CSE 5306 - DISTRIBUTED SYSTEMS

PROJECT 1

REPORT

REMOTE PROCEDURE CALL BASED COMMUNICATION

Note: I have neither given nor received unauthorized assistance on this work

Signed: ALETI SHIVANI REDDY **Signed: SAI KRISHNA REDDY**
Date: 03-03-2022

NAME: SAI KRISHNA REDDY SEELAM

E-Mail ID: sxs3838@mavs.uta.edu

NetID: sxs3838

ID: 1002303838

NAME: SHIVANI REDDY ALETI

E-Mail ID: sxa4487@mavs.uta.edu

NetID: sxa4487

ID:1002034487

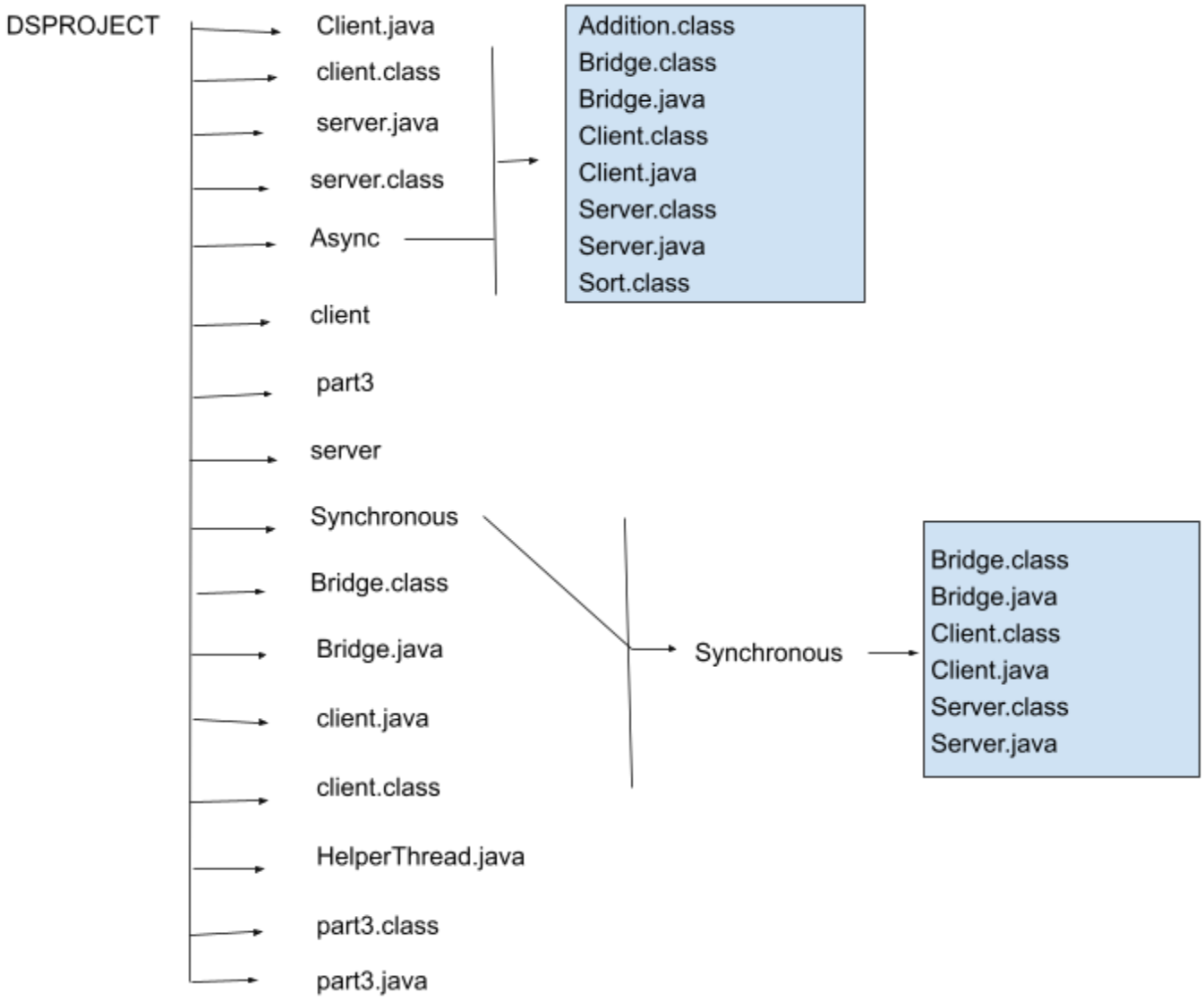
OVERALL STATUS

When assigned to this project, we had basic knowledge on RPC. We took a glance at the project, then understood the problem statements and started working on it. Initially, we face few challenges to write code and get the files uploaded. But for that, we have gone through some websites, books and got a better idea of what communication protocols.

We were able to write the code in Java and deploy protocols to the server. Synchronization has been the biggest task for us in the project. We tried our best to learn and develop the best in java, but we got many errors initially. We finally developed the code for the given project and completed the task.

FILE DESCRIPTION

The files which are included : **DSPROJECT** folder which contains all the required files and folders which are required to execute this project.


DSPROJECT
Client.java
client.class
server.java
server.class
Async
client
part3
server
Synchronous
Bridge.class
Bridge.java
client.java
client.class
HelperThread.java
part3.class
part3.java
Addition.class
Bridge.class
Bridge.java
Client.class
Client.java
Server.class
Server.java
Sort.class
Synchronous
Bridge.class
Bridge.java
Client.class
Client.java
Server.class
Server.java

Division of Work

We both have basic knowledge when the project was assigned to us. Though We divided the work equally we executed the entire project on both of our PC's. We have divided the given parts of the project into four parts as given. Part 2 and part 3 was done by SaiKrishna and Part1, Part 4 was done by Shivani**.** Though we have done work individually we have shared our knowledge of what we have done and what methods and procedures were used in the project so that we both get knowledge equally.

ISSUES FACED AND WAYS TO APPROACH THEM

1. Project is hard to understand at one glance:

The given project has a lot of information and four parts to finish.

For this, We went question by question and noted down what should be done in which part of the question. And divided the tasks equally between us, we have figured out what websites, youtube videos to refer to.

2. Issues faced during the project :

-> After installing Ubuntu on vmware, we had to always shut it down and restart the virtual machine as we were facing screen freezing issues and a few other issues.

-> As client and server connection was a little tricky at the beginning by using rpc. We referred to few resources and understood how it works.

-> Understanding Synchronous , Asynchronous and deferred synchronous was a little confusing. After doing little research about these topics we were able to implement it.

PART 2 :

Extract the given compressed zip folder into a regular folder. Then extract the data as in the below steps:

after

Firstly, Import the given file DSProject into the java environment. Change the path in the Server.Java file which is used to import the code from the local folder. Copy path from where it has been saved in the local device and paste as required. Run the file **Server.Java** which is the code for server. It executes and displays the output as

```
shivani@shivani-VirtualBox:~$ cd Desktop
shivani@shivani-VirtualBox:~/Desktop$ ls
 Async            DSPROJECT        Synchronous
 async_CS         Evaluate.class  'TableExample$1.class'
'client server'   Evaluate.java    TableExample.class
 client_server_1  mian_cs          TableExample.java
 client_server_2  rpc
 cS               rpc2
shivani@shivani-VirtualBox:~/Desktop$ cd DSPROJECT
shivani@shivani-VirtualBox:~/Desktop/DSPROJECT$ LS
LS: command not found
shivani@shivani-VirtualBox:~/Desktop/DSPROJECT$ ls
Bridge.class  client.java        part3.java    test.txt
Bridge.java   HelperThread.class  server
client        part3              server.class
client.class  part3.class        server.java
shivani@shivani-VirtualBox:~/Desktop/DSPROJECT$ java server
Started Server....
```

Secondly, run the file **Client.Java** which has the code for client side with four operations namely, Upload, Delete, Rename, Download.

```
shivani@shivani-VirtualBox:~/Desktop/DSPROJECT$ java client
Enter the Type of Operation : upload or delete or rename or download
```

Next, Select an operation and type in as “upload” select the file and type in the name to upload the file into the server.

```
shivani@shivani-VirtualBox:~/Desktop/DSPROJECT$ java client
Enter the Type of Operation : upload or delete or rename or download upload
Enter Client File name
test.txt
upload is done.shivani@shivani-VirtualBox:~/Desktop/DSPROJECT$
```

Select “Download” and then give the file name to download the file to the client. The output is shown likewise:

```
shivani@shivani-VirtualBox:~/Desktop/DSPROJECT$ java client
Enter the Type of Operation : upload or delete or rename or download download
Enter Server File Name to Download
test.txt
download is done
```

Run once again and now select “rename” file as input command. It asks for Enter Current File Name: Give the current file name once this step is done it shows as Enter New File Name: Give the new name that you want. Output is like:

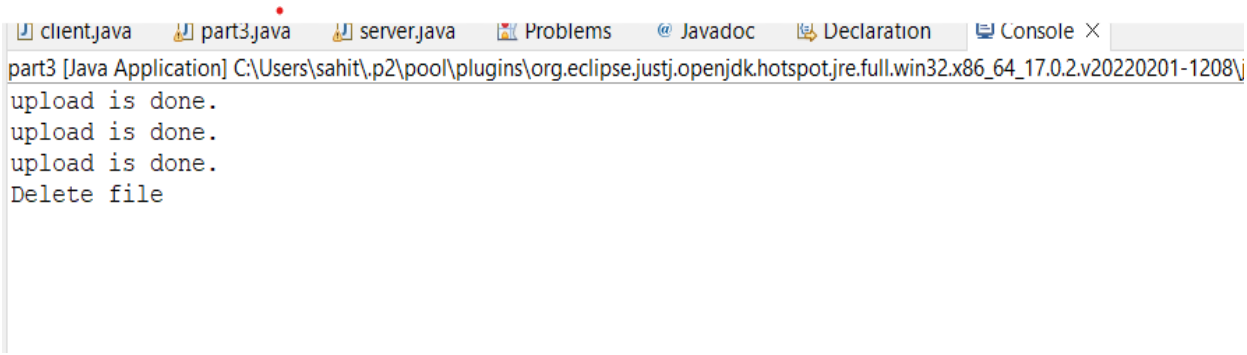
```
shivani@shivani-VirtualBox:~/Desktop/DSPROJECT$ java client
Enter the Type of Operation : upload or delete or rename or download rename
Enter Current File Name
test.txt
Enter New File Name
test1.txt
rename is done
```

Now select “delete” and the file you want to delete if the file is not found it shows an error:

```
shivani@shivani-VirtualBox:~/Desktop/DSPROJECT$ java client
Enter the Type of Operation : upload or delete or rename or download delete
Enter Server File Name
test.txt
Delete is failed
```

PART 3 :

Firstly, import the part3.Java file and execute the file. Since for the first time the code runs, the server will be considered as empty and all the files in the client will be updated in the server folder. For the next iteration(next periodic cycle). Code runs again and any changes in client will be reflected in server. When the file update, delete, or has been edited old files are stored in oldarray and new files are stored and updated in the server.



PART 4 :

Synchronous:

Download the Synchronous file and unzip it. Now try to run and compile **Server.java** file by using **javac Server.java && java server** command in ubuntu terminal.

```
shivani@shivani-VirtualBox:~$ cd Desktop
shivani@shivani-VirtualBox:~/Desktop$ ls
 Async             client_server_2   Evaluate.java   Synchronous
 async_CS          cS                mian_cs        'TableExample$1.class'
'client server'    DSPROJECT         rpc             TableExample.class
 client_server_1   Evaluate.class    rpc2            TableExample.java
shivani@shivani-VirtualBox:~/Desktop$ cd Synchronous/
shivani@shivani-VirtualBox:~/Desktop/Synchronous$ ls
Synchronous
shivani@shivani-VirtualBox:~/Desktop/Synchronous$ cd Synchronous
shivani@shivani-VirtualBox:~/Desktop/Synchronous/Synchronous$ ls
Bridge.class  Bridge.java  Client.class  Client.java  Server.class  Server.java
shivani@shivani-VirtualBox:~/Desktop/Synchronous/Synchronous$ javac Server.java && java Server
Started Server....
```

Now we have to open a new terminal and try to access **client.java**. Run and compile client.java by using **javac Client.java && java client** command in ubuntu terminal.

```
shivani@shivani-VirtualBox:~/Desktop/Synchronous/Synchronous$ javac Client.java && java Client
Please choose any one operation : 1-> Add or 2-> Sort or 3-> Exit
```

Now the client is expected to give an input for choosing any one of the operations. For example, if a client chooses addition he goes for option 1 and enters “1” in the terminal.

```
shivani@shivani-VirtualBox:~/Desktop/Synchronous/Synchronous$ javac Client.java && java Client
Please choose any one operation : 1-> Add or 2-> Sort or 3-> Exit
1
Addition of Two Integers
Enter first number:
```

After entering option ”1” it triggers the addition function in client.java first and performs certain operations as per request.
Now we have to enter the first number and second number to perform addition operation.

```
shivani@shivani-VirtualBox:~/Desktop/Synchronous/Synchronous$ javac Client.java && java Client
Please choose any one operation : 1-> Add or 2-> Sort or 3-> Exit
1
Addition of Two Integers
Enter first number:
3
Enter second number::
4
Result of adding the two numbers = 7
Please choose any one operation : 1-> Add or 2-> Sort or 3-> Exit
```

After generating the result , the client can choose or perform any other operation .

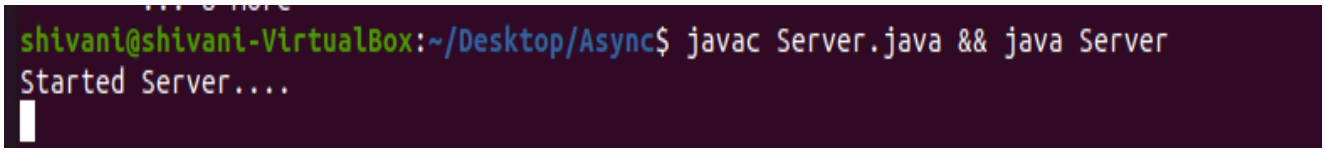
```
shivani@shivani-VirtualBox:~/Desktop/Synchronous/Synchronous$ javac Client.java && java Client
Please choose any one operation : 1-> Add or 2-> Sort or 3-> Exit
1
Addition of Two Integers
Enter first number:
3
Enter second number::
4
Result of adding the two numbers = 7
Please choose any one operation : 1-> Add or 2-> Sort or 3-> Exit
2
Sorting an array
Enter number of elements in the array:
4
3
5
1
2
Array elemets are:
3
5
1
2
Sorted array: 1 2 3 5
Please choose any one operation : 1-> Add or 2-> Sort or 3-> Exit
3
Program will now exit
```

Synchronous is something which schedules the task one by one so there is no overlapping of the process involved.

Asynchronous:

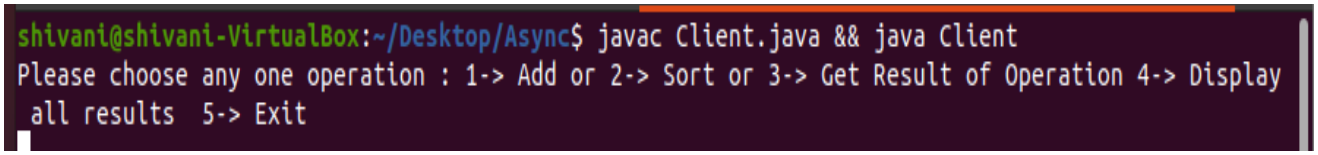
Let's get into Asynchronous processing.

Download the Asynchronous file and unzip it. Now try to run and compile Server.java file by using javac Server.java && java server command in ubuntu terminal.



Now we have to open a new terminal and try to access client.java. Run and compile client.java

by using javac Client.java && java client command in ubuntu terminal.



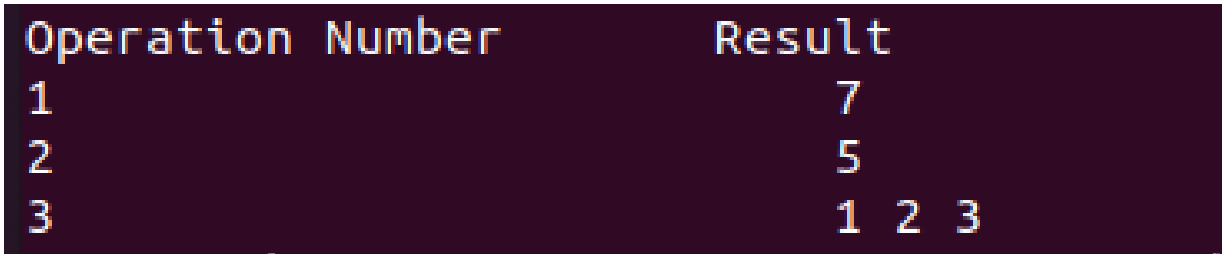
Now the client is expected to give an input for choosing any one of the operations . For example, if a client chooses addition he goes for option 1 and enters “11” in the terminal.

```
shivani@shivani-VirtualBox:~/Desktop/Async$ javac Client.java && java Client
Please choose any one operation : 1-> Add or 2-> Sort or 3-> Get Result of Operation 4-> Display
 all results  5-> Exit
1
Addition of Two Integers
Enter first number:
2
Enter second number::
3
Please choose any one operation : 1-> Add or 2-> Sort or 3-> Get Result of Operation 4-> Display
 all results  5-> Exit
2
Sorting an array
Enter number of elements in the array:
3
Enter elements of the array:
3
2
1
Please choose any one operation : 1-> Add or 2-> Sort or 3-> Get Result of Operation 4-> Display
 all results  5-> Exit
4
Operation Number       Result
1                          7
2                          5
3                          1 2 3
Please choose any one operation : 1-> Add or 2-> Sort or 3-> Get Result of Operation 4-> Display
 all results  5-> Exit
3
```

At first the client is trying to perform addition operation and then he wants to perform some other operation immediately without knowing the result of the first one. When the second operation is performed the first task is already done by the time the client receives the result for the second task.

By this we can understand that asynchronous tasks are something which allows you to send multiple requests at a time and then can render whichever output you need of that particular task from the server.

The results or response is been stored in the table



The first two rows in the table represents the results of add operation and the 3 rd row represents the result of sorting an array.