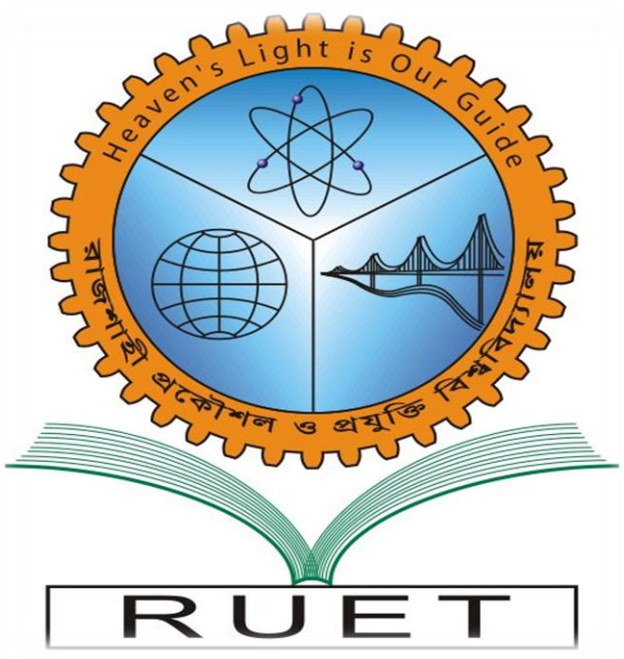

Lab Report 3

Course Title : Sessional Based on CSE 3205
Course No. : CSE 3206

Submitted by

Name: Md. Shabir Khan Akash.

Class: 3rd year, Even semester.

Department : CSE

Roll: 1603108

Section: B

Submitted to

Dr. A. H. M. Sarowar Sattar

Professor

Department of CSE, RUET

Objective:

To understand the stop and wait strategy (sending 10 objects) in a server-client program.

Description :

Step 1 : A java file was created named **MyServerClassObject.java** which was a server program.

Step 2: A class was declared named **ParityObject** in that java file using the implements keyword as serializable. “Serializable” is a marker interface (has no data member and method). It is used to “mark” java classes so that objects of these classes may get certain capability.

Step 3: In the **ParityObject** class **header**, **data**, **protocol**, **tailer** were declared as variable where their data type was **int**, **String**, **int**, **int** respectively. protocol was initialized with 10. Then a function was declared named **ParityObject()** having the parameters of integer, String, integer and integer. In that function, this pointer was used to assign the value from the main function to the variable of **ParityObject** class. Then a **showDetails()** function was declared to show the output of the program.

Step 4: In the main function under public class **MyServerClassObject**, ServerSocket, Socket objects were declared to establish or initialize a serversocket and socket. Then an **ObjectOutputStream** was declared to get the object that will be passed to the client program. Then a **BufferedReader** was declared to take the string data as a buffer stream in the input. Then a Byte type array and StringBuilder was declared to convert the string into binary and a counter was used to count the number of 1s in that binary bitstream. And then calculating the total number of 1s including header and data if there was odd number of 1s, then tailer was assigned with 1 else 0. After that, **ParityObject()** function was called in the main function and then **writeObject()** method was used to write the **ObjectOutputStream** that was declared earlier. Then an **ObjectInputStream** was declared to get the object that will be coming from the client program for acknowledge. Then **ParityObject** type object was declared and **readObject()** method was used on that object. Then **output()** method was also used on that same object to get the output in the server program. This process was done inside a for loop that executes 10 times. So 10 objects can be sent to client program.

Step 5: Then A java file was created named **MyClientClassObject.java** which was a client program.

Step 6: In the main function under public class **MyClientClassObject**, Socket object was declared to establish or initialize a socket. Then an **ObjectInputStream** was declared to get the object that will be coming from the server program. Then **ParityObject** type object was declared and **readObject()** method was used on that object. Then **showDetails()** method was also used on that same object to get the output.

Step 7: A class was declared named **ParityObject** in that java file using the implements keyword as serializable. In the **ParityObject** class **header**, **data**, **protocol**, **tailer** were declared as variable where their data type was **int**, **String**, **int**, **int** respectively. protocol was initialized with 10. Then a function was declared named **ParityObject()** having the parameters of integer, String, integer and integer. In that function, this pointer was used to assign the value from the main function to the variable of **ParityObject** class. Then a **output()** function was declared to show the output of the program. **ParityObject** type object was declared where the header holds the acknowledge value and the rest of the parameters were null and then **writeObject()** method was used to send that object containing the acknowledge value to the server program.

Step 8: At the end, when the loop was executed for 10^{th} time, then **close()** method was used both in **MyServerClasObject.java** and **MyClientClassObject.java** to close the socket and disconnect the connection between server and client.

Conclusion:

From the above procedure, 10 objects having some variables from server to client can be passed to client program and to check the error, stop and wait strategy was used to watch whether there was any change in the acknowledge on the server side as well as client side. And in this program, after receiving the object from server, client will send an object to server if the acknowledge is okay. That object will print the acknowledge that holds the header. So, this is stop and wait strategy.

Appendix

MyServerClassObject.java

```
import java.net.*;
import java.io.*;
import java.io.Serializable;
import java.util.*;
import java.lang.*;

class ParityObject implements Serializable
{
        int header;
        String data;
        int protocol = 10;
        int tailer;
```

```
   ParityObject(int header, String data, int protocol, int tailer)
   {
      this.header = header;
      this.data = data;
      this.protocol = protocol;
      this.tailer = tailer;
   }

         void showDetails()
         {
                  System.out.println("Header was: "+header);
                  System.out.println("Entered Data String was: "+data);
                  System.out.println("protocol was: "+protocol);
                  System.out.println("Tailer is: "+tailer);
                  System.out.println("Parity Matched !!! So, Acknowledge = Header = "+header);
         }
}

public class MyServerClassObject
{
   public static void main(String arg[]) throws Exception
   {
         int ct = 0;
         ServerSocket server=new ServerSocket(1700);
            Socket s=server.accept();
                  System.out.println("******Connection Established******\n");

                  ObjectOutputStream os = new ObjectOutputStream(s.getOutputStream());
                  ObjectInputStream iss = new ObjectInputStream(s.getInputStream());
                  BufferedReader buffRead = new BufferedReader(new InputStreamReader(System.in));

         for (int i1=0; i1<10 ;i1++ ) {
                  System.out.println("------------------------------------------------------------");
                        System.out.println("\nEnter Header and Data String respectively: \n");
                        Scanner sc = new Scanner(System.in);
                        int header = sc.nextInt();
                        String data = buffRead.readLine();
                        byte[] bytes = data.getBytes();
                        StringBuilder binary = new StringBuilder();
                        int count = 0;
                        int pro = 1;
                        int protocol = 10;
                        int tailer = 0;

                        for (byte b : bytes)
                        {
                                 int val = b;
                                 for (int i = 0; i < 8; i++)
                                 {
                                          binary.append((val & 128) == 0 ? 0 : 1);
                                          if((val & 128) != 0)
                                                   count++;
                                          val <<= 1;
                                 }
                                 binary.append(' ');
                        }
```

```
                        int head=0;

                        if(header == 1 || header == 2 || header == 4 || header == 8 || header == 10)
                                head = 1;
                        if(header == 3 || header == 5 || header == 6 || header == 9)
                                head = 2;
                        if(header == 7)
                                head = 3;

                        if(((pro+head+count) % 2) == 0)
                                tailer += 0;
                        else
                                tailer += 1;


                    ParityObject object1 = new ParityObject(header,data,protocol,tailer);
                    os.writeObject(object1);
                    ct++;

                    ParityObject pop = (ParityObject)iss.readObject();
                        System.out.println("------------------------------------------------------------");
                        pop.output();

                        if(ct == 10)
                                s.close();
            }
    }
}
```

MyClientClassObject.java

```
import java.io.*;
import java.lang.*;
import java.util.*;
import java.net.*;
import java.io.Serializable;

class ParityObject implements Serializable
{
        int header;
        String data;
        int protocol = 10;
        int tailer;

    ParityObject(int header, String data, int protocol, int tailer)
    {
        this.header = header;
        this.data = data;
        this.protocol = protocol;
        this.tailer = tailer;
    }
    void showDetails()
    {
        System.out.println("Header was: "+header);
```

```
            System.out.println("Entered Data String was: "+data);
            System.out.println("protocol was: "+protocol);
            System.out.println("Tailer is: "+tailer);
            System.out.println("Parity Matched !!! So, Acknowledge = Header = "+header);
  }


            void output()
            {
                        System.out.println("Recieved object from client !!! Acknowledge = "+header);
            }
}
public class MyClientClassObject
{
   public static void main(String arg[]) throws Exception
   {
            int ct2=0;
      Socket s = new Socket("localhost",1700);
      ObjectOutputStream oos = new ObjectOutputStream(s.getOutputStream());
            ObjectInputStream is = new ObjectInputStream(s.getInputStream());
            int header=1;
            for(int j=0; j<10 ; j++) {
                                    ParityObject p = (ParityObject)is.readObject();
                                    System.out.println("------------------------------------------------------------");
                                    p.showDetails();
                                    ParityObject object11 = new ParityObject(header,null,0,0);
                        oos.writeObject(object11);
                        header++;

                                    ct2 ++;
            if(ct2 == 10)
                        s.close();
            }
   }
}
```