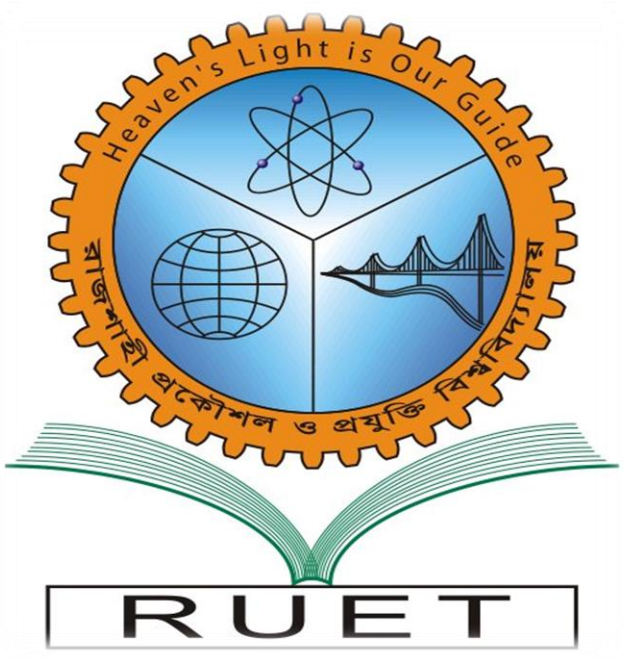

Lab Report 2

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 error correction strategy (parity check) in a server-client program.

Description :

Step 1 : A java file was created named **MyServerClass.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 **MyServerClass**, 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.

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

Step 6: In the main function under public class **MyClientClass**, 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: At the end, **close()** method was used to close the socket and disconnect the connection between server and client.

Conclusion:

From the above procedure, an object having some variables from server to client can be passed to client program and to check the error, parity checking was used to watch whether there was any change in the parity on the server side as well as client side. So, error checking was done.

Appendix

MyServerClass.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("\nEntered Data String was: "+data);
		System.out.println("\nprotocol was: "+protocol);
		System.out.println("\nTailer is: "+tailer);
		System.out.println("\nParity Matched");

	}
}

public class MyServerClass
{
   public static void main(String arg[]) throws Exception
   {
	ServerSocket server=new ServerSocket(1700);
	Socket s=server.accept();

		System.out.println("*****Connection Established*****\n");


		System.out.println("Enter Header and Data String respectively: \n");

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

		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(' ');
                }

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



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

        s.close();
   }
}
```

MyClientClass.java

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

public class MyClientClass
{
   public static void main(String arg[]) throws Exception
   {
      Socket s = new Socket("localhost",1700);
      ObjectInputStream is = new ObjectInputStream(s.getInputStream());
      ParityObject p = (ParityObject)is.readObject();
      p.showDetails();
      s.close();
   }
}
```