Title: Implementation of shared memory

1) Shared Memory
   Code:
   SharedMemoryClient.java

```
package lab_06.ShareMemory;

import java.io.*;
import java.net.*;

public class SharedMemoryClient {

        public static void main(String args[]) throws Exception {
                BufferedReader sin;
                PrintStream sout;
                BufferedReader stdin;
                Socket sk = new Socket(InetAddress.getLocalHost(), 2000);
                sin = new BufferedReader(new
InputStreamReader(sk.getInputStream()));
                sout = new PrintStream(sk.getOutputStream());
                stdin = new BufferedReader(new InputStreamReader(System.in));
                String s;
                while (true) {
                        System.out.println("Client: ");
                        s = stdin.readLine();
                        sout.println(s);
                        s = sin.readLine();
                        System.out.println("Server replied: " + s);
                        break;
                }
                sin.close();
                sout.close();
                stdin.close();
        }
}
```

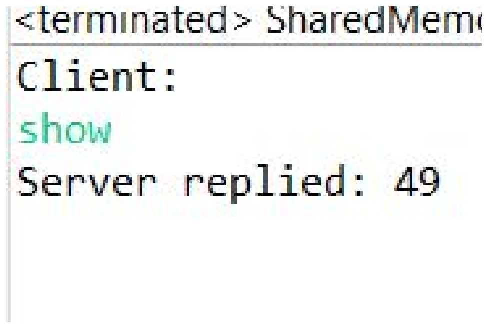
SharedMemoryServer.java

```
package lab_06.ShareMemory;
```

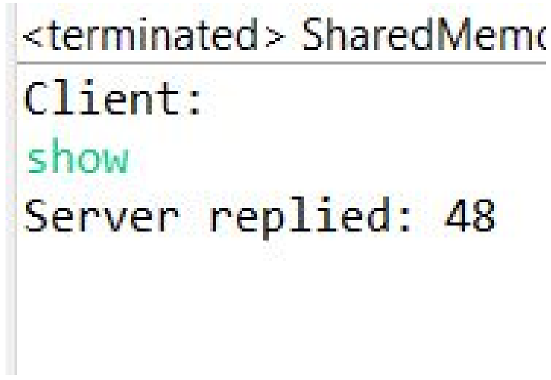
```
import java.io.*;
import java.net.*;
import java.util.*;

public class SharedMemoryServer {

        static int a = 50;
        static int count = 0;

        public static void getA(PrintStream cout) {
                count++;
                --a;
                cout.println(a);
        }

        public void setA(int a) {
                SharedMemoryServer.a = a;
        }

        public static void main(String args[]) throws Exception {
                String op;
                ServerSocket ss = new ServerSocket(2000);
                while (true) {
                        Socket sk = ss.accept();
                        BufferedReader cin = new BufferedReader(new
InputStreamReader(sk.getInputStream()));
                        PrintStream cout = new PrintStream(sk.getOutputStream());
                        System.out.println("Client request from" +
sk.getInetAddress().getHostAddress() + " accept");
                        BufferedReader stdin = new BufferedReader(new
InputStreamReader(System.in));
                        String s;
                        s = cin.readLine();
                        Scanner sc = new Scanner(s);
                        op = sc.next();
                        if (op.equalsIgnoreCase("show")) {
                                getA(cout);
                        } else {
                                cout.println("Check Syntax");
```

```
                    break;
                }
                System.out.println("Count: " + count);
                sk.close();

                cin.close();
                cout.close();
                stdin.close();
                sc.close();
            } // close while loop ss.close();
            ss.close();
        }

    }
```

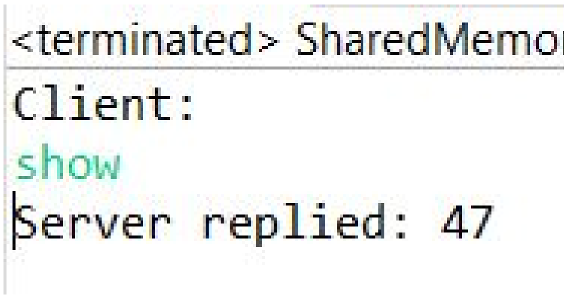
Screenshot:



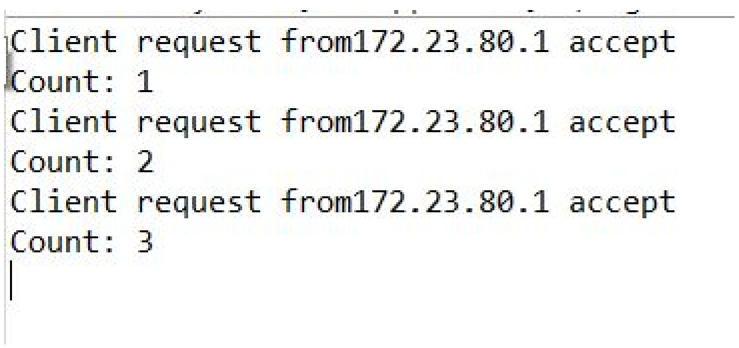
```
<terminated> SharedMem
Client:
show
Server replied: 49
```



```
<terminated> SharedMem
Client:
show
Server replied: 48
```



```
<terminated> SharedMemo
Client:
show
Server replied: 47
```



```
Client request from172.23.80.1 accept
Count: 1
Client request from172.23.80.1 accept
Count: 2
Client request from172.23.80.1 accept
Count: 3
```

2) Load Balancing
Code:
LoadBalancer.java

```
package lab_06.LoadBalancing;

import java.net.DatagramPacket;
import java.net.DatagramSocket;

/** * RPCServer_Date */
public class LoadBalancer {
        static DatagramSocket serverDatagramSocket;
        static DatagramPacket clientDataPacket;
        static byte buf[];
        static int s1 = 0, s2 = 5;
        static int s2PORT = 5002, s1PORT = 5001;

        public static void main(String[] args) {
                try {
                        System.out.println("Load Balancer Daemon up");
                        buf = new byte[1024];
                        serverDatagramSocket = new DatagramSocket(5000);
                        clientDataPacket = new DatagramPacket(buf, buf.length);
                        while (true) {
```

```
                                        serverDatagramSocket.receive(clientDataPacket);
                                        int PORTtoSend = 0;
                                        String currTime = new
String(clientDataPacket.getData(), 0, clientDataPacket.getLength());
                                        byte[] operationRes = currTime.getBytes();
                                        if (s1 > s2) {
                                                PORTtoSend = s2PORT;
                                                s2++;
                                        } else {
                                                PORTtoSend = s1PORT;
                                                s1++;

                                        }
                                        DatagramPacket resDataPacket = new
DatagramPacket(operationRes, operationRes.length,
                                                        clientDataPacket.getAddress(),
PORTtoSend);
                                        serverDatagramSocket.send(resDataPacket);
                                        System.out.println("Sent packet to server at: " +
resDataPacket.getPort());
                                }
                        } catch (Exception e) {
                                e.printStackTrace();
                        } finally {
                                serverDatagramSocket.close();
                        }
                }
}
```

Client.java

```
package lab_06.LoadBalancing;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class Client {
	static DatagramPacket msgDatagramPacket;
	static DatagramSocket clientDatagramSocket;
	static byte[] data;
	static BufferedReader br;
	static int LB_PORT = 5000;

	public static void main(String[] args) {
		try {
			br = new BufferedReader(new
InputStreamReader(System.in));
			data = new byte[1024];

			clientDatagramSocket = new DatagramSocket();
			while (true) {
				System.out.println("Msg: ");
				String msg = br.readLine();
				data = msg.getBytes();
				msgDatagramPacket = new DatagramPacket(data,
data.length, InetAddress.getByName("localhost"), LB_PORT);
				clientDatagramSocket.send(msgDatagramPacket);
				System.out.println("Packet sent to server at PORT: "
+

						LB_PORT);

				Thread.sleep(1000);
			}

		} catch (Exception e) {
			e.printStackTrace();
		} finally {
			clientDatagramSocket.close();
		}
	}
}
```

MyServerOne.java

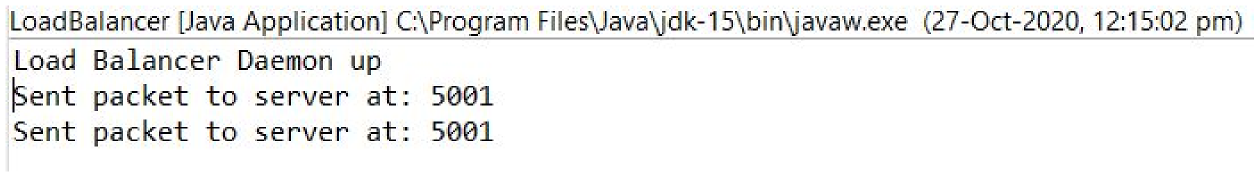
```
package lab_06.LoadBalancing;
```

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;

/** * RPCServer_Date */
public class MyServerOne {
	static DatagramSocket serverDatagramSocket;
	static DatagramPacket clientDataPacket;
	static byte buf[];
	static int PORT = 5001;

	public static void main(String[] args) {
		try {
			System.out.println("Waiting for client packet...");
			buf = new byte[1024];
			serverDatagramSocket = new DatagramSocket(PORT);
			clientDataPacket = new DatagramPacket(buf, buf.length);
			while (true) {
				serverDatagramSocket.receive(clientDataPacket);
				String res = new String(clientDataPacket.getData(), 0,
clientDataPacket.getLength());
				System.out.println("Received: " + res);
			}
		} catch (Exception e) {
			e.printStackTrace();
		} finally {
			serverDatagramSocket.close();
		}
	}
}
```

MyServerTwo.java

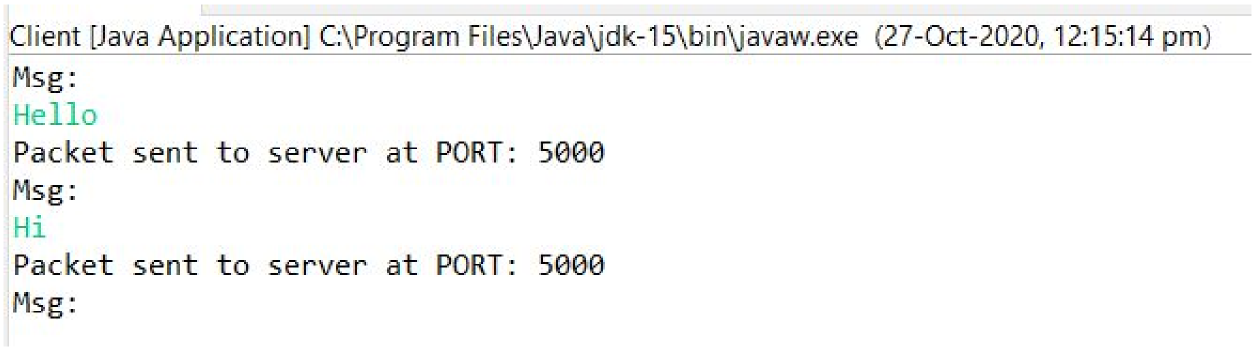
```
package lab_06.LoadBalancing;

import java.net.DatagramPacket;
import java.net.DatagramSocket;

/** * RPCServer_Date */
public class MyServerTwo {
	static DatagramSocket serverDatagramSocket;
	static DatagramPacket clientDataPacket;
	static byte buf[];
	static int PORT = 5002;

	public static void main(String[] args) {
		try {
			System.out.println("Waiting for client packet...");
			buf = new byte[1024];
			serverDatagramSocket = new DatagramSocket(PORT);
			clientDataPacket = new DatagramPacket(buf, buf.length);
			while (true) {
				serverDatagramSocket.receive(clientDataPacket);
				String res = new String(clientDataPacket.getData(), 0,
clientDataPacket.getLength());
				System.out.println("Received: " + res);
			}
		} catch (Exception e) {
			System.out.println(e.toString());
		} finally {
			serverDatagramSocket.close();
			PORT++;
		}
	}
}
```

Screenshot:



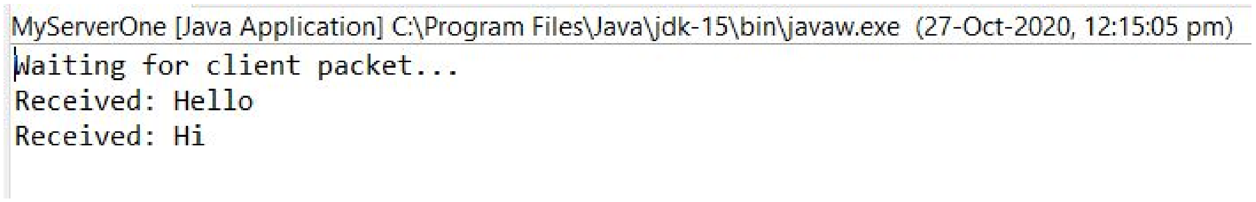
LoadBalancer [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (27-Oct-2020, 12:15:02 pm)

```
Load Balancer Daemon up
Sent packet to server at: 5001
Sent packet to server at: 5001
```



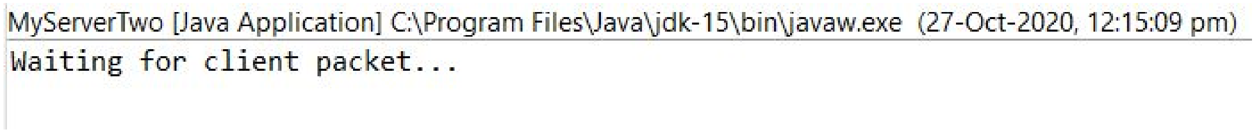
Client [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (27-Oct-2020, 12:15:14 pm)

```
Msg:
Hello
Packet sent to server at PORT: 5000
Msg:
Hi
Packet sent to server at PORT: 5000
Msg:
```



MyServerOne [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (27-Oct-2020, 12:15:05 pm)

```
Waiting for client packet...
Received: Hello
Received: Hi
```



MyServerTwo [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe (27-Oct-2020, 12:15:09 pm)

```
Waiting for client packet...
```