Sanyam Agrawal SE21UCSE192 CSE3

DS - Assignment 4

4.1->

SERVER Programs:

SO_KEEPALIVE.java :->

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class SO_KEEPALIVE {
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(8080);

            System.out.println("Server is running. Waiting for a client to
connect...");

            Socket clientSocket = serverSocket.accept();

            // Enable SO_KEEPALIVE
            clientSocket.setKeepAlive(true);
            // Test SO_KEEPALIVE
            System.out.println("SO_KEEPALIVE enabled: " +
clientSocket.getKeepAlive());

            // Close the sockets
            clientSocket.close();
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

SO_LINGER.java :->

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
public class SoLinger {
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(8080);
            System.out.println("Server is running. Waiting for a client to
connect...");
            Socket clientSocket = serverSocket.accept();

            // Enable SO_LINGER with a timeout of 30 seconds on the client
socket
            clientSocket.setSoLinger(true, 30);
            System.out.println("SO_LINGER enabled: " +
clientSocket.getSoLinger());

            // Close the sockets
            clientSocket.close();
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

SO_SNDBUF_and_SO_RCVBUF.java :->

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class SO_SNDBUF_and_SO_RCVBUF {
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(8080);

            System.out.println("Server is running. Waiting for a client to
connect...");

            Socket clientSocket = serverSocket.accept();
            // Set SO_SNDBUF and SO_RCVBUF to custom values (e.g., 8192 bytes)
```

```
            clientSocket.setSendBufferSize(8192);
            clientSocket.setReceiveBufferSize(8192);

            // Test SO_SNDBUF and SO_RCVBUF
            System.out.println("SO_SNDBUF: " +
clientSocket.getSendBufferSize());
            System.out.println("SO_RCVBUF: " +
clientSocket.getReceiveBufferSize());

            // Close the sockets
            clientSocket.close();
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

TCPNODELAY.java :->

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class TCPNODELAY {
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(8080);

            System.out.println("Server is running. Waiting for a client to
connect...");

            Socket clientSocket = serverSocket.accept();

            // Disable TCP_NODELAY
            clientSocket.setTcpNoDelay(false);

            // Test TCP_NODELAY
            System.out.println("TCP_NODELAY enabled: " +
!clientSocket.getTcpNoDelay());

            // Close the sockets
            clientSocket.close();
```

```
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Client Program:

ClientSocketTest.java :->

```
import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;

public class ClientSocketTest {
    public static void main(String[] args) {
        String serverAddress = "localhost";
        int serverPort = 8080;

        try {
            // Create a socket and connect to the server
            Socket socket = new Socket(serverAddress, serverPort);
            System.out.println("Connected to server " + serverAddress + " on
port " + serverPort);

            // Keep the connection open for 10 seconds
            try {
                Thread.sleep(10000); // 10 seconds
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            // Close the socket
            socket.close();
            System.out.println("Disconnected from server.");
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
}
```

OUTPUT:

SO_KEEPALIVE.java :->

```
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> javac .\SO_KEEPALIVE.java
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> java SO_KEEPALIVE
Server is running. Waiting for a client to connect...
SO_KEEPALIVE enabled: true
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4>
```

```
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> javac .\ClientSocketTest.java
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> java ClientSocketTest
Connected to server localhost on port 8080
Disconnected from server.
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4>
```

SO_LINGER.java :->

```
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> javac .\SoLinger.java
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> java SoLinger
Server is running. Waiting for a client to connect...
SO_LINGER enabled: 30
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4>
```

```
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> javac .\ClientSocketTest.java
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> java ClientSocketTest
Connected to server localhost on port 8080
Disconnected from server.
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4>
```

SO_SNDBUF_and_SO_RCVBUF.java :->

```
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> javac .\SO_SNDBUF_and_SO_RCVBUF.java
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> java SO_SNDBUF_and_SO_RCVBUF
Server is running. Waiting for a client to connect...
SO_SNDBUF: 8192
SO_RCVBUF: 8192
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4>
```

```
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> javac .\ClientSocketTest.java
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> java ClientSocketTest
Connected to server localhost on port 8080
Disconnected from server.
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4>
```

TCPNODELAY.java :->

```
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> javac .\TCPNODELAY.java
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> java TCPNODELAY
Server is running. Waiting for a client to connect...
TCP_NODELAY enabled: true
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4>
```

```
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> javac .\ClientSocketTest.java
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> java ClientSocketTest
Connected to server localhost on port 8080
Disconnected from server.
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4>
```

4.2->

SharedMemoryExample.java:->

```java
import java.util.concurrent.locks.ReentrantLock;

class SharedCounter {
    private int counter = 0;
    private ReentrantLock lock = new ReentrantLock();

    // Method to safely increment the counter
    public void increment() {
        lock.lock(); // Acquire the lock
        try {
            counter++; // Increment the shared counter
```

```
        } finally {
            lock.unlock(); // Ensure the lock is released even if an exception
occurs
        }
    }

    // Method to get the current counter value
    public int getValue() {
        return counter;
    }
}

class IncrementTask implements Runnable {
    private SharedCounter sharedCounter;

    public IncrementTask(SharedCounter sharedCounter) {
        this.sharedCounter = sharedCounter;
    }

    @Override
    public void run() {
        for (int i = 0; i < 1000; i++) {
            sharedCounter.increment(); // Safely increment the counter
        }
    }
}

public class SharedMemoryExample {
    public static void main(String[] args) throws InterruptedException {
        System.out.println("Sanyam Agrawal SE21UCSE192");


        SharedCounter sharedCounter = new SharedCounter();

        // Create multiple threads
        Thread[] threads = new Thread[10];
        for (int i = 0; i < threads.length; i++) {
            threads[i] = new Thread(new IncrementTask(sharedCounter));
            threads[i].start(); // Start each thread
        }

        // Wait for all threads to finish
        for (Thread t : threads) {
            t.join();
        }
```

```
        // Print the final counter value
        System.out.println("Final counter value: " + sharedCounter.getValue());
    }
}
```

OUTPUT:

```
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> javac .\SharedMemoryExample.java
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4> java SharedMemoryExample
Sanyam Agrawal SE21UCSE192
Final counter value: 10000
PS C:\Users\pc\Desktop\SE21UCSE192_Dis_Sys\Lab4>
```