Department of ICT
Faculty of Technology
University of Ruhuna

ICT3253 – Distributed and Cloud Computing **Level 3 - Semester –**

2 Lab Sheet 07

Objectives:
Understanding of basic concepts of socket programming and practicing socket programming with TCP and UDP implementations.
Java should be used as the programming language for the implementations of distributed computing.

Part B

1. Write a program using **Java TCP socket programming** to build a communication between client and server (sending message from client to server and server to client).

```
package com.labsheetseven.b.server;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class ServerTCP {
   public static void main(String[] args) throws IOException {
       ServerSocket serverSocket = new ServerSocket(6666);
       System.out.println("Server is waiting for client request...");
       Socket clientSocket = serverSocket.accept();
       System.out.println("Client connected.");
       BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
       PrintWriter out = new PrintWriter(clientSocket.getOutputStream(),
true);
       String clientMessage;
       while ((clientMessage = in.readLine()) != null) {
```

```
            System.out.println("Client: " + clientMessage);
            out.println("Received your message: " + clientMessage);
        }
        serverSocket.close();
    }
}
```

```
package com.labsheetseven.b.client;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class ClientTCP {
    public static void main(String[] args) throws IOException {
        Socket clientSocket = new Socket("localhost", 6666);
        PrintWriter out = new PrintWriter(clientSocket.getOutputStream(),
true);
        BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
        BufferedReader stdIn = new BufferedReader(new
InputStreamReader(System.in));
        String userInput;
        while ((userInput = stdIn.readLine()) != null) {
            out.println(userInput);
            System.out.println("Server: " + in.readLine());
        }
        clientSocket.close();
    }
}
```

2. Write a program using **Java UDP socket programming** to build a communication between client and server (sending message from client to server and server to client).

```java
package com.labsheetseven.b.server;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class ServerUDP {
    public static void main(String[] args) throws IOException {
        DatagramSocket socket = new DatagramSocket(4445);
        byte[] buffer = new byte[256];
        DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
        while (true) {
            socket.receive(packet);
            String received = new String(packet.getData(), 0, packet.getLength());
            System.out.println("Client: " + received);
            if (received.equals("end")) {
                break;
            }
        }
        socket.close();
    }
}
```

```java
package com.labsheetseven.b.client;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class ClientUDP {
    public static void main(String[] args) throws IOException {
        DatagramSocket socket = new DatagramSocket();
        InetAddress address = InetAddress.getByName("localhost");
        byte[] buf;
        buf = "Hello, server!".getBytes();
        DatagramPacket packet = new DatagramPacket(buf, buf.length, address, 4445);
        socket.send(packet);
        packet = new DatagramPacket(buf, buf.length);
        socket.receive(packet);
        String received = new String(packet.getData(), 0, packet.getLength());
        System.out.println("Server: " + received);
        socket.close();
    }
}
```

Part C

Transferring a file using socket programming.

- Create a text file in one of the local disks in your computer. Save the text file with the name “original.txt”.
- Use **Java TCP socket programming** to transfer the “original.txt” to another local disk in your computer using client and server communication with sockets.
- The transferred file name should be “copy.txt”.
- Finally observe the data inside the “copy.txt” file. What are your observations?
  file has successfully copied to the destination.

```
package com.labsheetseven.c.server;

import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
   public static void main(String[] args) throws IOException {
       ServerSocket servsock = new ServerSocket(12122);
       File myFile = new File("D:\\original.txt");
       while (true) {
           Socket sock = servsock.accept();
           byte[] mybytearray = new byte[(int) myFile.length()];
           BufferedInputStream bis = new BufferedInputStream(new
FileInputStream(myFile));
           bis.read(mybytearray, 0, mybytearray.length);
           OutputStream os = sock.getOutputStream();
           os.write(mybytearray, 0, mybytearray.length);
           os.flush();
           sock.close();
       }
   }
}
```

```
package com.labsheetseven.c.client;

import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.net.Socket;

public class Client {
   public static void main(String[] args) throws Exception {
       Socket sock = new Socket("127.0.0.1", 12122);
       byte[] mybytearray = new byte[1024];
       InputStream is = sock.getInputStream();
       FileOutputStream fos = new FileOutputStream("E:\\copy.txt");
       BufferedOutputStream bos = new BufferedOutputStream(fos);
       int bytesRead = is.read(mybytearray, 0, mybytearray.length);
```

```
        bos.write(mybytearray, 0, bytesRead);
        bos.close();
        sock.close();
    }
}
```

Part D

Ceylon Medtech Company is a leading seller of medical products and equipment Island wide. The company has headquarters located in Colombo and branches spread covering all the cities. Price changes and the discounts are decided and managed by the main branch (headquarters). All the time, requests from branches are coming to the main branch (headquarters) to get the final price of a product. Currently they manage this by making a call over the phone. But now it is difficult to manage this manual process because customer orders are increased. The higher management decides to develop a small software to get final prices and discounts from the headquarters to the branches when branches request for them.

The following list indicates their requirements:

- An application/program needs to run 24 hours at the headquarters.
- The program can be considered as a price storing server which keeps all the prices and discounts in maps.
- In a Price map, the map items are added to have the keys as the item code. It keeps the item initial price as the value (It is the initial price before deducting the discount).
- In a Discount map, the map items are added to have the keys as the item code. It keeps the discount of item as the value.

- All branches island-wide to connect to the server (Headquarters) to get the final price of an item by sending the item code and getting the Initial price (before deducting discount), Discount and Final price (after deducting discount) as result from the server (Headquarters).

Consider the following sample data (Table 01 and 02) and adjust your program according to it.

Item Code	Initial Price in lkr (Before deducting discount)
MT001	2500.00
MT002	1200.00
MT003	350.00
MT004	990.00
ES001	4000.00
ES002	3400.00
ES003	6500.00
ES004	1500.00
MLS001	750.00
MLS002	4500.00

Table 01

Item Code	Discount (%)
MT001	5
MT002	2.5
MT003	0
MT004	2
ES001	10
ES002	7.5
ES003	15
ES004	5
MLS001	0
MLS002	10

Table 02

Suppose the company has hired you to develop the software/program. You are asked to do the following:

a) Write a program using **Java TCP socket programming**. Use streams properly. Your program should contain both server – for headquarters and clients – for branches. Your program should get followings as the results from the server after sending the item code to the server.
   - Initial price (Before deducting discount)
   - Discount value
   - Final price (After deducting discount)

```
package com.labsheetseven.d.server;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashMap;

public class ServerTCP {
   public static void main(String[] args) throws IOException {
```

```
        ServerSocket servsock = new ServerSocket(12121);

        HashMap<String, Double> prices = new HashMap<String, Double>() {{
            put("MT001", 2500.00);
            put("MT002", 1200.00);
            put("MT003", 350.00);
            put("MT004", 990.00);
            put("ES001", 4000.00);
            put("ES002", 3400.00);
            put("ES003", 6500.00);
            put("ES004", 1500.00);
            put("MLS0001", 750.00);
            put("MLS0002", 4500.00);
        }};

        HashMap<String, Double> discounts = new HashMap<String, Double>() {{
            put("MT001", 5.0);
            put("MT002", 2.5);
            put("MT003", 0.0);
            put("MT004", 2.0);
            put("ES001", 10.0);
            put("ES002", 7.5);
            put("ES003", 15.0);
            put("ES004", 5.0);
            put("MLS0001", 0.0);
            put("MLS0002", 10.0);
        }};

        while (true) {
            Socket sock = servsock.accept();
            BufferedReader in = new BufferedReader(new
InputStreamReader(sock.getInputStream()));
            PrintWriter out = new PrintWriter(sock.getOutputStream(), true);

            String itemCode = in.readLine();
            Double initialPrice = prices.get(itemCode);
            Double discount = discounts.get(itemCode);
            Double finalPrice = initialPrice * (1 - discount / 100);

            out.println("Initial price: " + initialPrice);
            out.println("Discount (%): " + discount);
            out.println("Final price: " + finalPrice);

            sock.close();
        }
    }
}
```

```java
package com.labsheetseven.d.client;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class ClientTCP {
   public static void main(String[] args) throws Exception {
       Socket sock = new Socket("127.0.0.1", 12121);
       BufferedReader in = new BufferedReader(new
InputStreamReader(sock.getInputStream()));
       PrintWriter out = new PrintWriter(sock.getOutputStream(), true);

       String itemCode = "MLS0002";
       out.println(itemCode);

       System.out.println(in.readLine());
       System.out.println(in.readLine());
       System.out.println(in.readLine());

       sock.close();
   }
}
```

```
Initial price: 4500.0
Discount (%): 10.0
Final price: 4050.0

Process finished with exit code 0
```

b) Implement the above scenario again with **Java UDP socket programming.**

```java
package com.labsheetseven.d.server;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.util.HashMap;

public class ServerUDP {
   public static void main(String[] args) throws IOException {
       DatagramSocket ds = new DatagramSocket(12121);

       HashMap<String, Double> prices = new HashMap<String, Double>() {{
           put("MT001", 2500.00);
```

```
            put("MT002", 1200.00);
            put("MT003", 350.00);
            put("MT004", 990.00);
            put("ES001", 4000.00);
            put("ES002", 3400.00);
            put("ES003", 6500.00);
            put("ES004", 1500.00);
            put("MLS0001", 750.00);
            put("MLS0002", 4500.00);
        }};

        HashMap<String, Double> discounts = new HashMap<String, Double>() {{
            put("MT001", 5.0);
            put("MT002", 2.5);
            put("MT003", 0.0);
            put("MT004", 2.0);
            put("ES001", 10.0);
            put("ES002", 7.5);
            put("ES003", 15.0);
            put("ES004", 5.0);
            put("MLS0001", 0.0);
            put("MLS0002", 10.0);
        }};

        byte[] receive = new byte[65535];

        while (true) {
            DatagramPacket DpReceive = new DatagramPacket(receive,
receive.length);
            ds.receive(DpReceive);

            String itemCode = data(receive).toString();
            Double initialPrice = prices.get(itemCode);
            Double discount = discounts.get(itemCode);
            Double finalPrice = initialPrice * (1 - discount / 100);

            String result = "Initial price: " + initialPrice + "\nDiscount: "
+ discount + "\nFinal price: " + finalPrice;
            DatagramPacket DpSend = new DatagramPacket(result.getBytes(),
result.getBytes().length, DpReceive.getAddress(), DpReceive.getPort());
            ds.send(DpSend);

            receive = new byte[65535];
        }
    }

    public static StringBuilder data(byte[] a) {
        if (a == null)
            return null;
        StringBuilder ret = new StringBuilder();
        int i = 0;
        while (a[i] != 0) {
            ret.append((char) a[i]);
            i++;
        }
        return ret;
    }
}
```

```java
package com.labsheetseven.d.client;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class ClientUDP {
   public static void main(String[] args) throws IOException {
       DatagramSocket ds = new DatagramSocket();
       InetAddress ip = InetAddress.getLocalHost();
       byte buf[] = null;

       String itemCode = "MT001";
       buf = itemCode.getBytes();

       DatagramPacket DpSend = new DatagramPacket(buf, buf.length, ip,
12121);
       ds.send(DpSend);

       byte[] receive = new byte[65535];
       DatagramPacket DpReceive = new DatagramPacket(receive,
receive.length);
       ds.receive(DpReceive);

       System.out.println("Prices from server:-\n" + data(receive));
   }

   public static StringBuilder data(byte[] a) {
       if (a == null)
           return null;
       StringBuilder ret = new StringBuilder();
       int i = 0;
       while (a[i] != 0) {
           ret.append((char) a[i]);
           i++;
       }
       return ret;
   }
}
```

```
C:\Users\niraj\.jdks\openjdk-20.0.2\bin\java.exe
Prices from server:-
Initial price: 2500.0
Discount: 5.0
Final price: 2375.0

Process finished with exit code 0
```

Part E

Create a lab report including all the exercises in Part **B,C** and **D** rename it using your index number (Ex: Lab_07_Report_TGXXX). Submit it together with all the program codes that you implemented in exercises in Part **B, C and D** to the given submission link in the LMS.