United States International University

MIS 6060: DISTRIBUTED COMPUTING & INTERNET TECHNOLOGY

Lab Exercise 6 – Implementing a One Clock Synchronization Algorithm

Objective

To show the implementation of a one clock synchronization algorithm.

Requirements

The Net Beans IDE or Eclipse IDE can be used to demonstrate this exercise.

The program contains a client process and one server which host the synchronized Clocks. The client process randomly sends messages to the server. The server maintains a log of Messages and times at which they were sent. The message should finally display the messages accepted, discarded and the valid times (G) at the moment of receipt of messages.

Theory

Two clocks are said to be synchronized at a particular instance of time if the clock skew of the two clocks is less than some specified constant δ.

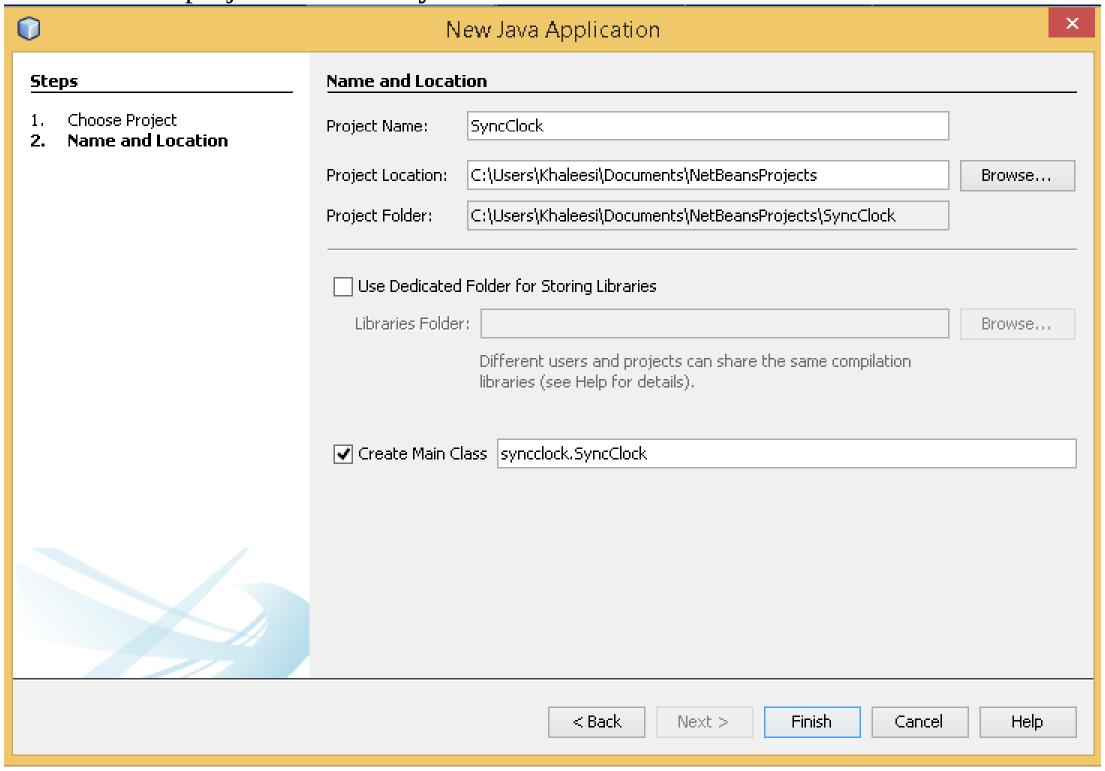
Every message carries a connection identifier and timestamp. For each connection, the server records in a table the most recent timestamp it has seen.

In the algorithm every message carries a connection identifier (chosen by the sender) and a timestamp. For each connection, the server records in the table the most recent timestamp it has seen. If an incoming message is lower than the timestamp stored for the connection, the message is rejected. The server continuously maintains global variable,

G = CurrentClockTime – MaxLifeTime – MaxClockSkew

Step 1: Create the Project

Create a new project. Name it **SyncClock**.


New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: SyncClock

Project Location: C:\Users\Khaleesi\Documents\NetBeansProjects

Project Folder: C:\Users\Khaleesi\Documents\NetBeansProjects\SyncClock

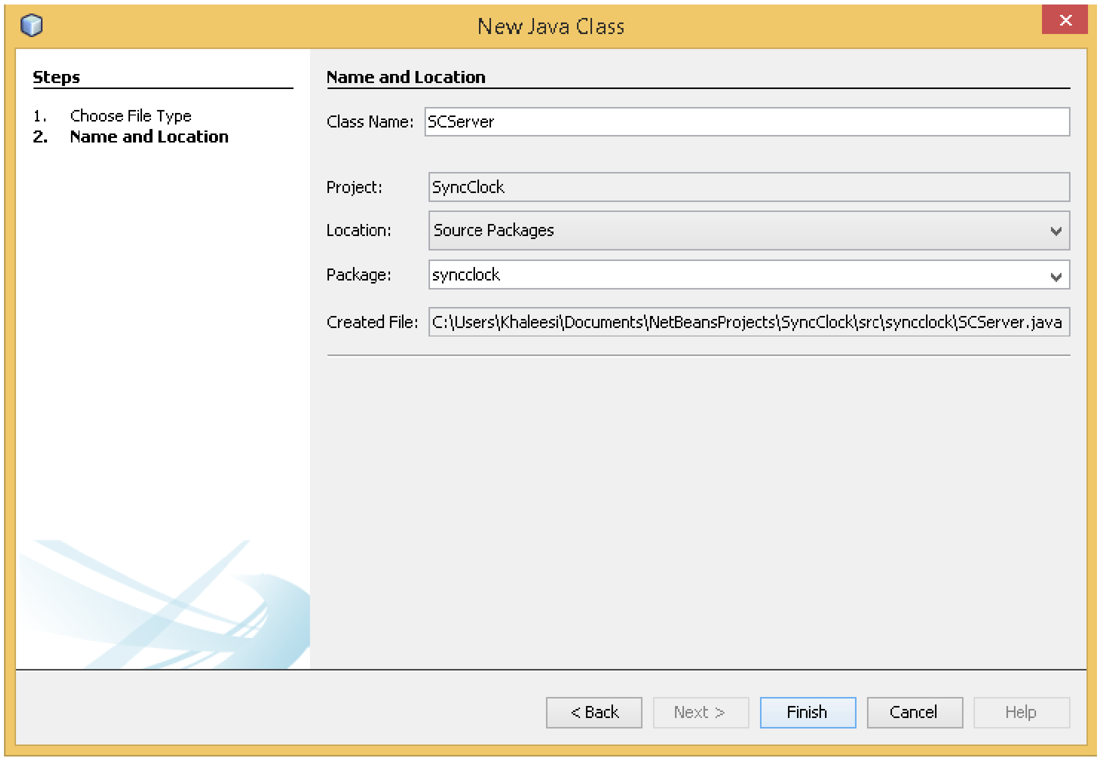
☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☑ Create Main Class syncclock.SyncClock

Create two new classes: ***SCServer.java*** and ***SCClient.java***



Name and Location

Class Name: SCServer

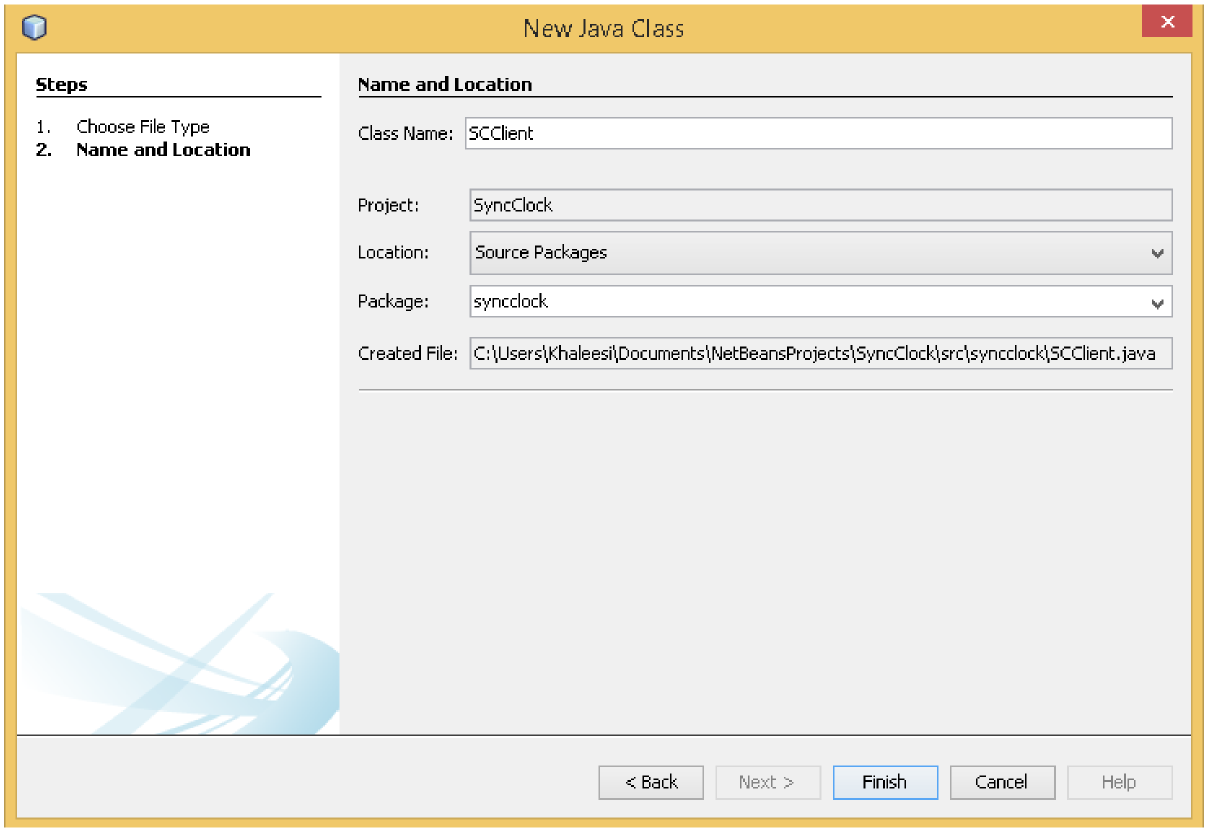
Project: SyncClock

Location: Source Packages

Package: syncclock

Created File: C:\Users\Khaleesi\Documents\NetBeansProjects\SyncClock\src\syncclock\SCServer.java

and ***SCClient.java***


New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: SCClient

Project: SyncClock

Location: Source Packages

Package: syncclock

Created File: C:\Users\Khaleesi\Documents\NetBeansProjects\SyncClock\src\syncclock\SCClient.java

Step 2

Insert code to the ***SCServer*** class you created

SCServer.java code

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

public class SCServer
{           public static void main(String args[])throws
Exception
            {
            InetAddress lclhost;
            lclhost=InetAddress.getLocalHost();

                        long maxtime,skewtime,datatime;
            String maxtimestr,skewtimestr;
            BufferedReader br;
            ClntServer ser=new ClntServer(lclhost);

                        System.out.println("Enter the maximum time");
br = new BufferedReader(new InputStreamReader(System.in));
maxtimestr=br.readLine();
                        System.out.println("Enter the maximum skew time");
br = new BufferedReader(new InputStreamReader(System.in));
skewtimestr=br.readLine();

                        maxtime=Long.parseLong(maxtimestr);
            skewtime=Long.parseLong(skewtimestr);

            while(true)                  {
datatime = System.currentTimeMillis();
long G = datatime-maxtime-skewtime;
System.out.println("G ="+G);
ser.setTimeStamp(new Timestamp(G));
ser.recPort(8001);
                                    ser.recData();
            }
            }
}

class ClntServer
{
            InetAddress
lclhost;            int
recport;
            Timestamp obtmp;

            ClntServer(InetAddress lclhost)
            {
            this.lclhost = lclhost;
            }

            void recPort(int recport)
            {
            this.recport = recport;
            }

            void setTimeStamp(Timestamp obtmp)
            {
this.obtmp = obtmp;
            }
```

```
        void recData()throws Exception
        {
        String msgstr="";
        DatagramSocket ds;
        DatagramPacket dp;
BufferedReader br;
        byte buf[] = new byte[256];

        ds = new DatagramSocket(recport);
dp = new DatagramPacket(buf,buf.length);
ds.receive(dp);
        ds.close();

        msgstr = new String(dp.getData(),0,dp.getLength());
        System.out.println(msgstr);

       Timestamp obtmp = new Timestamp(Long.parseLong(msgstr));

        if(this.obtmp.before(obtmp) == true)
        {
                        System.out.println("The Message is accepted");
        }
else
        {
          System.out.println("The Message is rejected");
        }
        }
}
```

Step 3: Insert code to the ***SCClient*** Class you created

SCClient.java code

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

public class SCClient
{
        public static void main(String args[])throws Exception
        {
        InetAddress lclhost;
        lclhost=InetAddress.getLocalHost();

                while(true)
        {
```

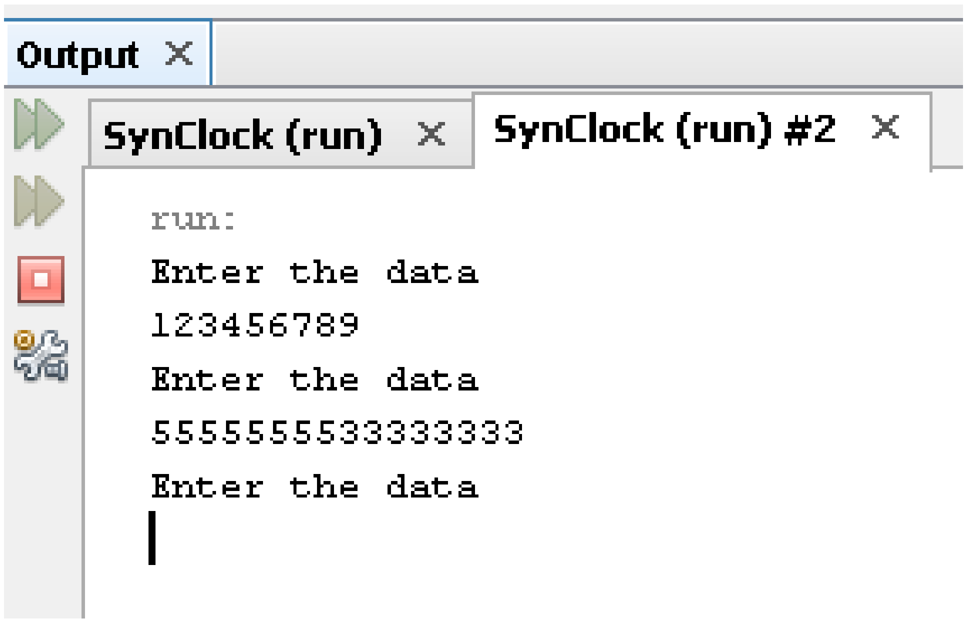
```
                                        Client cntl=new
Client(lclhost);
cntl.sendPort(9001);
cntl.sendData();
        }
        }
}

class Client
{
        InetAddress lclhost;
        int senport;

        Client(InetAddress lclhost)
        {
        this.lclhost=lclhost;
        }

        void sendPort(int senport)
        {
        this.senport=senport;
        }

        void sendData()throws Exception
        {
        DatagramPacket dp;
        DatagramSocket ds;
BufferedReader br;
        br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the
data");         String str=br.readLine();
ds = new DatagramSocket(senport);
        dp = new DatagramPacket(str.getBytes(),str.length(),lclhost,senport-
1000);          ds.send(dp);            ds.close();
        }
}
```

Step 4

<u>Output</u>:

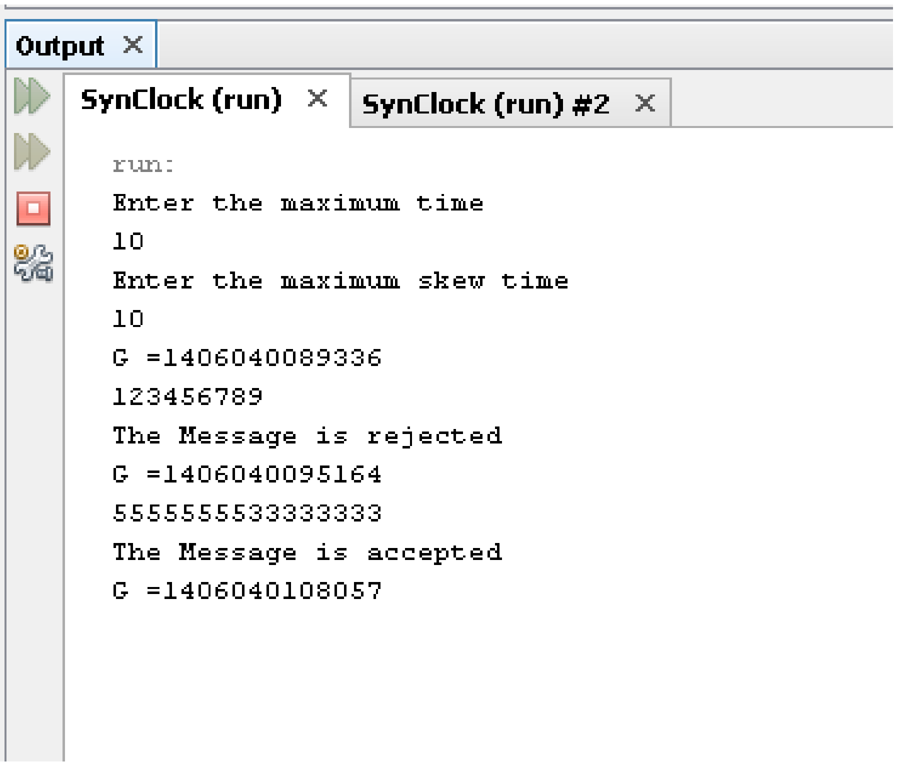
Run the ***SCServer.java*** file first then run the ***SCClient.java*** file. The output should be as follows on the client depending on the times you entered when prompted for Maximum time and Skew time;

Scenario 1: One Client



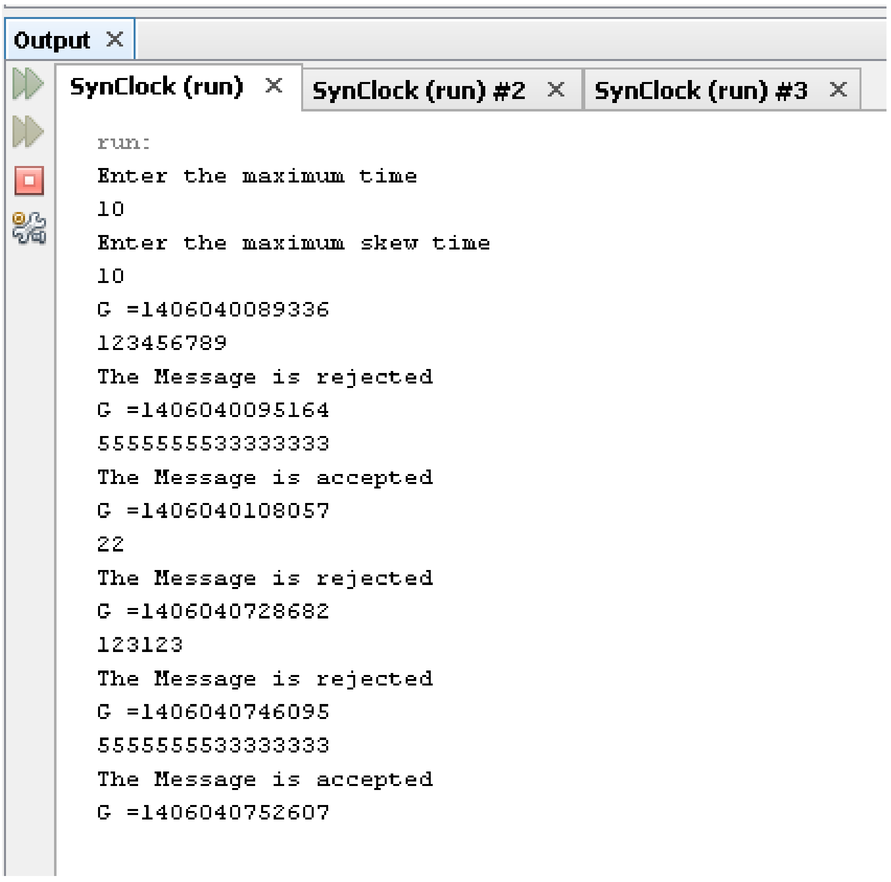
```
run:
Enter the data
123456789
Enter the data
5555555533333333
Enter the data
```

Observe the server Output should display similar to below depending on what numbers you have entered



```
run:
Enter the maximum time
10
Enter the maximum skew time
10
G =1406040089336
123456789
The Message is rejected
G =1406040095164
5555555533333333
The Message is accepted
G =1406040108057
```

Scenario 2: more than one client
Check the output on server output when multiple clients run



```
run:
Enter the maximum time
10
Enter the maximum skew time
10
G =1406040089336
123456789
The Message is rejected
G =1406040095164
5555555533333333
The Message is accepted
G =1406040108057
22
The Message is rejected
G =1406040728682
123123
The Message is rejected
G =1406040746095
5555555533333333
The Message is accepted
G =1406040752607
```