DSL ASSIGNMENT-1

CODE

ClientImplementation.java

```
package Client;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

/**
 * This Java class implements the ClientInterface and defines a method to receive and print a
message.
 */
public class ClientImplementation extends UnicastRemoteObject implements ClientInterface {
    protected ClientImplementation() throws RemoteException {
    }

    public void receiveMessage(String message) throws RemoteException {
        System.out.println("Received message: " + message);
    }
}
```

ClientInterface.java

```
package Client;
import java.rmi.Remote;
import java.rmi.RemoteException;

// This code is defining a remote interface called `ClientInterface` that extends the `Remote`
// interface. It declares a single method called `receiveMessage` that takes a `String` parameter
and
// throws a `RemoteException`. This interface is used to define the methods that can be called
remotely
// by a server in a distributed system.
public interface ClientInterface extends Remote {
    void receiveMessage(String message) throws RemoteException;
}
```

ClientMain.java

```
package Client;
```

```
import java.rmi.Naming;
import java.util.Scanner;
import Server.ServerInterface;
/**
 * This Java class registers a client with a server and broadcasts a message to all registered
clients.
 */

public class ClientMain {
    public static void main(String[] args) {
        try {
            ServerInterface server = (ServerInterface) Naming.lookup("rmi://localhost/Server");
            ClientInterface client = new ClientImplementation();
            server.registerClient(client);

            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter a message to broadcast: ");
            String message = scanner.nextLine();

            server.broadcastMessage(message);
        } catch (Exception e) {
            System.out.println("Client exception: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Note: Put all these client files in a folder named Client

ServerImplementation.java

```
package Server;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;
import java.util.List;

/**
 * This Java class implements a server interface that allows clients to register and broadcast
messages
 * to all registered clients.
 */
```

```
public class ServerImplementation extends UnicastRemoteObject implements ServerInterface {
    private List<Client.ClientInterface> clients;

    public ServerImplementation() throws RemoteException {
        clients = new ArrayList<>();
    }

    public void registerClient(Client.ClientInterface client) throws RemoteException {
        clients.add(client);
    }

    public void broadcastMessage(String message) throws RemoteException {
        System.out.println("Broadcasting message: " + message);
        for (Client.ClientInterface client : clients) {
            client.receiveMessage(message);
        }
    }
}
```

ServerInterface.java

```
package Server;
import java.rmi.Remote;
import java.rmi.RemoteException;

import Client.ClientInterface;

// This is a Java interface for a remote server that extends the `Remote` interface, indicating that
it
// can be accessed remotely. It declares two methods that can be called by clients:
`registerClient`
// and `broadcastMessage`.
public interface ServerInterface extends Remote {
    void registerClient(ClientInterface client) throws RemoteException;
    void broadcastMessage(String message) throws RemoteException;
}
```

ServerMain.java

```
package Server;
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
```

```java
/**
 * This Java class starts a server using RMI (Remote Method Invocation) technology.
 */

public class ServerMain {
    public static void main(String[] args) {
        try {
            ServerInterface server = new ServerImplementation();
            LocateRegistry.createRegistry(1099);
            Naming.rebind("rmi://localhost/Server", server);
            System.out.println("Server started.");
        } catch (Exception e) {
            System.out.println("Server exception: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Note: Put all these server files in a folder named Server

STEPS:

1. **Terminal 1**
   javac Client/*.java
   javac Server/*.java
   rmiregistry 5000

2. **Terminal 2**
   java Server.ServerMain

3. **Terminal 3**
   java Client.ClientMain

Terminal 1

```
C:\Users\hp\Desktop\a1>javac Client/*.java

C:\Users\hp\Desktop\a1>javac Server/*.java

C:\Users\hp\Desktop\a1>rmiregistry 5000
```

Terminal 2

```
PS C:\Users\hp\Desktop\a1> java Server.ServerMain
Server started.
Broadcasting message: HELLO
```

Terminal 3

```
PS C:\Users\hp\Desktop\a1> java Client.ClientMain
Enter a message to broadcast: HELLO
Received message: HELLO
```

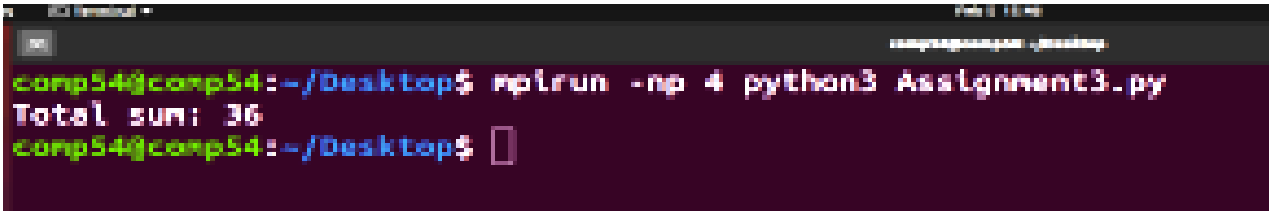
DSL ASSIGNMENT-3

CODE

```
from mpi4py import MPI
import numpy as np

def distribute_elements(array, comm):
	n = len(array)
	local_n = n // comm.Get_size()  # Compute the local size of the array
	local_array = np.empty(local_n, dtype=int)  # Create an empty array to store the local
elements
	comm.Scatter(array, local_array, root=0)  # Scatter the elements from the root process to
all processes
	local_sum = np.sum(local_array)  # Compute the local sum of the elements
	return local_sum

if __name__ == '__main__':
	comm = MPI.COMM_WORLD
	rank = comm.Get_rank()  # Get the rank of the current process

	if rank == 0:
	array = np.array([1, 2, 3, 4, 5, 6, 7, 8])  # Specify an array with 8 elements (divisible by 4
processes)
	else:
	array = None

	array = comm.bcast(array, root=0)  # Broadcast the array from the root process to all
processes

	local_sum = distribute_elements(array, comm)  # Compute the local sum of elements for
each process
	all_sums = comm.gather(local_sum, root=0)  # Gather all the local sums to the root
process

	if rank == 0:
	total_sum = np.sum(all_sums)  # Compute the total sum by adding all the individual
sums
	print("Total sum:", total_sum)  # Print the total sum
```

STEPS

1. pip install numpy
2. pip install mpi4py --upgrade
3. mpirun -np 4 python3 Assignment3.py



DSL ASSIGNMENT-4

CODE

```
import java.util.List;
import java.util.ArrayList;
class TimeServer {
private List<Integer> clocks;
public TimeServer(List<Integer> clocks) {
this.clocks = clocks;
}
public void synchronizeClocks() {
int sum = 0;
int average;
// Calculate the sum of all clocks
for (int clock : clocks) {
sum += clock;
}
// Calculate the average clock time
average = sum / clocks.size();
// Adjust each clock to the average time
for (int i = 0; i < clocks.size(); i++) {
clocks.set(i, average);
}
}
public List<Integer> getClocks() {
return clocks;
}
public static void main(String[] args) {
// Create a list of clocks with their initial times
List<Integer> clocks = new ArrayList<>();
clocks.add(100);
clocks.add(200);
clocks.add(150);
clocks.add(180);
// Create a time server with the clocks
TimeServer timeServer = new TimeServer(clocks);
// Synchronize the clocks using the Berkeley algorithm
timeServer.synchronizeClocks();
// Get the synchronized clocks
List<Integer> synchronizedClocks = timeServer.getClocks();
// Print the synchronized clocks
System.out.println("Synchronized Clock Times:");
```

```
for (int clock : synchronizedClocks) {
System.out.println(clock);
}
}
}
```

STEPS

1. sudo apt install openjdk-11-jre-headless
2. java assignment4.java

```
bcl14@bcl14:~/Desktop$ java assignment4.java
Synchronized Clock Times:
157
157
157
157
bcl14@bcl14:~/Desktop$
```

DSL ASSIGNMENT-5

CODE

```
import java.util.*;
import java.util.concurrent.TimeUnit;
class tokenring {
public static void main(String args[]) throws Throwable {
Scanner scan = new Scanner(System.in);
//Get and print all nodes
System.out.println("Enter the num of nodes:");
int nodes = scan.nextInt();
for (int i = 0; i < nodes; i++) {
System.out.print(" " + i);
}
System.out.println(" " + 0);
//Get sender, reciever and data, and initialize token to node 0
int token = 0;
int sender, reciever;
System.out.println("Enter sender:");
sender = scan.nextInt();
System.out.println("Enter receiver:");
reciever = scan.nextInt();
System.out.println("Enter Data:");
String data = scan.next();
//Keep passing the token until sender is found
System.out.print("Token passing:");
for (
int i = token, j = token;
(i % nodes) != sender;
i++, j = (j + 1) % nodes
) {
System.out.print(" " + j + "->");
TimeUnit.SECONDS.sleep(1);
}
System.out.println(" " + sender);
System.out.println(
"--------------------TOKEN WITH SENDER NOW PASSING DATA--------------------"
);
System.out.println("Sender " + sender + " sending data: " + data);
for (int i = sender + 1; i != reciever + 1; i = (i + 1) % nodes) {
System.out.print("data " + i + "->");
TimeUnit.SECONDS.sleep(1);
```

```
}
System.out.println();
System.out.println(
"------------------Receiver " +
reciever +

" received data: " +
data +
"---------------------\nodes"
);
token = sender;
scan.close();
}
}
```

STEPS

1. java assignment5.java

```
bcl14@bcl14:~/Desktop$ java assignment5.java
Enter the num of nodes:
3
 0 1 2 0
Enter sender:
2
Enter receiver:
0
Enter Data:
testing
Token passing: 0-> 1-> 2
--------------------TOKEN WITH SENDER NOW PASSING DATA------------------
Sender 2 sending data: testing
data 3->
-------------------Receiver 0 received data: testing---------------------
odes
bcl14@bcl14:~/Desktop$
```

DSL ASSIGNMENT-6

CODE

Bully.java

```
import java.util.*;

public class Bully {
    int coordinator;
    int max_processes;
    boolean processes[];


    public Bully(int max) {
        max_processes = max;
        processes = new boolean[max_processes];
        coordinator = max;

        System.out.println("Creating processes..");
        for(int i = 0; i < max; i++) {
            processes[i] = true;
            System.out.println("P"+ (i+1) + " created");
        }
        System.out.println("Process P" + coordinator + " is the coordinator");

    }
    void displayProcesses() {
        for(int i = 0; i < max_processes; i++) {
            if(processes[i]) {
                System.out.println("P" + (i+1) + " is up");
            } else {
                System.out.println("P" + (i+1) + " is down");
            }
        }
        System.out.println("Process P" + coordinator + " is the coordinator");
    }

    void upProcess(int process_id) {
        if(!processes[process_id - 1]) {
            processes[process_id - 1] = true;
            System.out.println("Process " + process_id + " is now up.");
        } else {
```

```
            System.out.println("Process " + process_id + " is already up.");
        }
    }

    void downProcess(int process_id) {
        if(!processes[process_id - 1]) {
            System.out.println("Process " + process_id + " is already down.");
        } else {
            processes[process_id - 1] = false;
            System.out.println("Process " + process_id + " is down.");
        }
    }

    void runElection(int process_id) {
        coordinator = process_id;
        boolean keepGoing = true;

        for(int i = process_id; i < max_processes && keepGoing; i++) {
            System.out.println("Election message sent from process " + process_id + " to process "
+ (i+1));

            if(processes[i]) {
                keepGoing = false;
                runElection(i + 1);
            }
        }
    }

    public static void main(String args[]) {
        Bully bully = null;
        int max_processes = 0, process_id = 0;
        int choice = 0;
        Scanner sc = new Scanner(System.in);

        while(true) {
            System.out.println("Bully Algorithm");
            System.out.println("1. Create processes");
            System.out.println("2. Display processes");
            System.out.println("3. Up a process");
            System.out.println("4. Down a process");
            System.out.println("5. Run election algorithm");
            System.out.println("6. Exit Program");
            System.out.print("Enter your choice:- ");
            choice = sc.nextInt();
```

```
            switch(choice) {
                case 1:
                    System.out.print("Enter the number of processes:- ");
                    max_processes = sc.nextInt();
                    bully = new Bully(max_processes);
                    break;
                case 2:
                    bully.displayProcesses();
                    break;
                case 3:
                    System.out.print("Enter the process number to up:- ");
                    process_id = sc.nextInt();
                    bully.upProcess(process_id);
                    break;
                case 4:
                    System.out.print("Enter the process number to down:- ");
                    process_id = sc.nextInt();
                    bully.downProcess(process_id);
                    break;
                case 5:
                    System.out.print("Enter the process number which will perform election:- ");
                    process_id = sc.nextInt();
                    bully.runElection(process_id);
                    bully.displayProcesses();
                    break;
                case 6:
                    System.exit(0);
                    break;
                default:
                    System.out.println("Error in choice. Please try again.");
                    break;
            }
        }
    }
}
```

Ring.java

```
import java.util.*;

public class Ring {
    int max_processes;
    int coordinator;
```

```
boolean processes[];
ArrayList<Integer> pid;

public Ring(int max) {
    coordinator = max;
    max_processes = max;
    pid = new ArrayList<Integer>();
    processes = new boolean[max];

    for(int i = 0; i < max; i++) {
        processes[i] = true;
        System.out.println("P" + (i+1) + " created.");
    }
    System.out.println("P" + (coordinator) + " is the coordinator");
}

void displayProcesses() {
    for(int i = 0; i < max_processes; i++) {
        if(processes[i])
            System.out.println("P" + (i+1) + " is up.");
        else
            System.out.println("P" + (i+1) + " is down.");
    }
    System.out.println("P" + (coordinator) + " is the coordinator");
}

void upProcess(int process_id) {
    if(!processes[process_id-1]) {
        processes[process_id-1] = true;
        System.out.println("Process P" + (process_id) + " is up.");
    } else {
        System.out.println("Process P" + (process_id) + " is already up.");
    }
}

void downProcess(int process_id) {
    if(!processes[process_id-1]) {
        System.out.println("Process P" + (process_id) + " is already down.");
    } else {
        processes[process_id-1] = false;
        System.out.println("Process P" + (process_id) + " is down.");
    }
}
```

```
    void displayArrayList(ArrayList<Integer> pid) {
        System.out.print("[ ");
        for(Integer x : pid) {
            System.out.print(x + " ");
        }
        System.out.print(" ]\n");
    }

    void initElection(int process_id) {
        if(processes[process_id-1]) {
            pid.add(process_id);

            int temp = process_id;

            System.out.print("Process P" + process_id + " sending the following list:- ");
            displayArrayList(pid);

            while(temp != process_id - 1) {
                if(processes[temp]) {
                    pid.add(temp+1);
                    System.out.print("Process P" + (temp + 1) + " sending the following list:- ");
                    displayArrayList(pid);
                }
                temp = (temp + 1) % max_processes;
            }
            coordinator = Collections.max(pid);
            System.out.println("Process P" + process_id + " has declared P" + coordinator + " as the
coordinator");
            pid.clear();
        }
    }

    public static void main(String args[]) {
        Ring ring = null;
        int max_processes = 0, process_id = 0;
        int choice = 0;
        Scanner sc = new Scanner(System.in);

        while(true) {
            System.out.println("Ring Algorithm");
            System.out.println("1. Create processes");
            System.out.println("2. Display processes");
            System.out.println("3. Up a process");
            System.out.println("4. Down a process");
```

```
            System.out.println("5. Run election algorithm");
            System.out.println("6. Exit Program");
            System.out.print("Enter your choice:- ");
            choice = sc.nextInt();

            switch(choice) {
                case 1:
                    System.out.print("Enter the total number of processes:- ");
                    max_processes = sc.nextInt();
                    ring = new Ring(max_processes);
                    break;
                case 2:
                    ring.displayProcesses();
                    break;
                case 3:
                    System.out.print("Enter the process to up:- ");
                    process_id = sc.nextInt();
                    ring.upProcess(process_id);
                    break;
                case 4:
                    System.out.print("Enter the process to down:- ");
                    process_id = sc.nextInt();
                    ring.downProcess(process_id);
                    break;
                case 5:
                    System.out.print("Enter the process which will initiate election:- ");
                    process_id = sc.nextInt();
                    ring.initElection(process_id);
                    break;
                case 6:
                    System.exit(0);
                    break;
                default:
                    System.out.println("Error in choice. Please try again.");
                    break;
            }
        }
    }
}
```

STEPS:

1. javac Bully.java
2. java Bully
3. javac Ring.java
4. java Ring

```
C:\Users\hp\Desktop\Assign6>javac Bully.java

C:\Users\hp\Desktop\Assign6>java Bully
Bully Algorithm
1. Create processes
2. Display processes
3. Up a process
4. Down a process
5. Run election algorithm
6. Exit Program
Enter your choice:- 1
Enter the number of processes:- 2
Creating processes..
P1 created
P2 created
Process P2 is the coordinator
Bully Algorithm
C:\Users\hp\Desktop\Assign6>javac Ring.java

C:\Users\hp\Desktop\Assign6>java Ring
Ring Algorithm
1. Create processes
2. Display processes
3. Up a process
4. Down a process
5. Run election algorithm
6. Exit Program
Enter your choice:- 1
Enter the total number of processes:- 2
P1 created.
P2 created.
P2 is the coordinator
Ring Algorithm
1. Create processes
2. Display processes
3. Up a process
4. Down a process
5. Run election algorithm
6. Exit Program
Enter your choice:- 2
P1 is up.
P2 is up.
P2 is the coordinator
Ring Algorithm
1. Create processes
2. Display processes
3. Up a process
4. Down a process
5. Run election algorithm
6. Exit Program
Enter your choice:- 3
Enter the process to up:- 1
Process P1 is already up.
Ring Algorithm
1. Create processes
2. Display processes
3. Up a process
4. Down a process
5. Run election algorithm
6. Exit Program
Enter your choice:- 4
Enter the process to down:- 2
Process P2 is down.
Ring Algorithm
1. Create processes
2. Display processes
3. Up a process
4. Down a process
5. Run election algorithm
6. Exit Program
Enter your choice:- 5
Enter the process which will initiate election:- 2
Ring Algorithm
1. Create processes
2. Display processes
3. Up a process
4. Down a process
5. Run election algorithm
6. Exit Program
Enter your choice:- |
```

DSL ASSIGNMENT-7

CODE

app.py

```
from flask import Flask

app = Flask(__name__)

@app.route('/hello')
def hello():
	return 'Hello, World!'

if __name__ == '__main__':
	app.run()
```

disapp.py

```
import requests

url = 'http://localhost:5000/hello'
response = requests.get(url)
print("Response from web service:", response.text)
```

STEPS:

1. pip install flask
2. Create app.py file
3. Run python3 app.py
4. Create a distributed_app.py file
5. Run python3 disapp.py
6. Open http://localhost:5000/hello for output.

```
bcl14@bcl14:~$  install Flask using pip install flask
install: target 'flask' is not a directory
bcl14@bcl14:~$  pip install flask
Collecting flask
  Downloading flask-3.0.3-py3-none-any.whl (101 kB)
     |████████████████████████████████| 101 kB 1.0 MB/s
Collecting itsdangerous>=2.1.2
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting click>=8.1.3
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
     |████████████████████████████████| 97 kB 4.1 MB/s
Collecting importlib-metadata>=3.6.0; python_version < "3.10"
  Downloading importlib_metadata-7.1.0-py3-none-any.whl (24 kB)
Collecting blinker>=1.6.2
  Downloading blinker-1.7.0-py3-none-any.whl (13 kB)
Collecting Werkzeug>=3.0.0
  Downloading werkzeug-3.0.2-py3-none-any.whl (226 kB)
     |████████████████████████████████| 226 kB 6.5 MB/s
Collecting Jinja2>=3.1.2
  Downloading Jinja2-3.1.3-py3-none-any.whl (133 kB)
     |████████████████████████████████| 133 kB 13.5 MB/s
Collecting zipp>=0.5
  Downloading zipp-3.18.1-py3-none-any.whl (8.2 kB)
Collecting MarkupSafe>=2.1.1
  Downloading MarkupSafe-2.1.5-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (26 kB)
Installing collected packages: itsdangerous, click, zipp, importlib-metadata, blinker, MarkupSafe, Werkzeug, Jinja2, flask
  WARNING: The script flask is installed in '/home/bcl14/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed Jinja2-3.1.3 MarkupSafe-2.1.5 Werkzeug-3.0.2 blinker-1.7.0 click-8.1.7 flask-3.0.3 importlib-metadata-7.1.0 itsdangerous-2.1.2 zipp-3.18.1
bcl14@bcl14:~$ http://localhost:5000/hello
bash: http://localhost:5000/hello: No such file or directory
bcl14@bcl14:~$ cd Desktop
bcl14@bcl14:~/Desktop$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [15/Apr/2024 13:48:55] "GET /hello HTTP/1.1" 200 -
127.0.0.1 - - [15/Apr/2024 13:49:04] "GET /hello HTTP/1.1" 200 -
127.0.0.1 - - [15/Apr/2024 13:49:04] "GET /favicon.ico HTTP/1.1" 404 -
```

es Terminal

bcl14@bcl14: ~/Desktop

```
bcl14@bcl14:~/Desktop$ python3 disapp.py
Response from web service: Hello, World!
bcl14@bcl14:~/Desktop$
```

localhost:5000/hello

Hello, World!