Assignment 1

AddserverIntf.java

```
import java.rmi.*;
public interface AddserverIntf extends Remote{
    double add(double d1, double d2)throws RemoteException;

}
```

Addserverimpl.java

```
import java.rmi.*;
import java.rmi.server.*;
public class Addserverimpl extends UnicastRemoteObject implements AddserverIntf{
    public Addserverimpl()throws RemoteException{}
    public double add(double d1, double d2)throws RemoteException{
        return d1 + d2;
    }
}
```

AddServer.java

```
import java.rmi.*;
public class AddServer {
    public static void main(String args[]){
        try{
            Addserverimpl obj = new Addserverimpl();
            Naming.rebind("AddServer", obj);
            System.out.println("Server is Ready ............");
        }catch(Exception e){
            System.out.println("Server Exception " + e);
        }
    }

}
```

AddClient.java

```
import java.rmi.*;
public class AddClient {
    public static void main(String[] args){
        try{
            String sreverurl = "rmi://localhost/AddServer";
            AddserverIntf addServer = (AddserverIntf)Naming.lookup(sreverurl);
            double d1 = 10.6;
            double d2 = 2.5;
            System.out.println("First Number : " + d1);
            System.out.println("Second Number : " + d2);
            System.out.println("Sum : " +addServer.add(d1, d2));
        }catch(Exception e){
            System.out.println("Client Exception" + e);
        }
    }

}
```

Assignment 4:

BerkeleyServer.java

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

public class BerkeleyServer {
    public static void main(String args[])throws Exception{
        ServerSocket ss =  new ServerSocket(5000);
        System.out.println("Waiting for Client........");
        Socket s = ss.accept();

        long serverTime = System.currentTimeMillis();
        BufferedReader in = new BufferedReader(new InputStreamReader(s.getInputStream()));
        long clientTime = Long.parseLong(in.readLine());



        long avgTime = (serverTime + clientTime) /2;
        PrintWriter out = new PrintWriter(s.getOutputStream(), true);
        out.println(avgTime);

        System.out.println("Server Time : " + new Date(serverTime));
        System.out.println("Client Time : " + new Date(clientTime));
        System.out.println("Sysnchronized Time : " + new Date(avgTime));

        s.close();
        ss.close();
    }
}
```

BerkeleyClient.java

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

public class BerkeleyClient {
    public static void main(String args[])throws Exception{
        Socket s = new Socket("127.0.0.1", 5000);

        long clientTime = System.currentTimeMillis();
        PrintWriter out = new PrintWriter(s.getOutputStream(),true);
        out.println(clientTime);

        BufferedReader in = new BufferedReader(new InputStreamReader(s.getInputStream()));
        long SyncTime = Long.parseLong(in.readLine());

        System.out.println("Client Time : " + new Date(clientTime));
        System.out.println("Synchronized Time : " + new Date(SyncTime));

        s.close();

    }
}
```

Assignment 5:

TokenRing.java

```
import java.util.*;
public class TokenRing {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter Number of Nodes : ");
        int n = sc.nextInt();

        System.out.println("Enter Sender Node : ");
        int sendernode = sc.nextInt();

        System.out.println("Enter Reciver Node : ");
        int recivernode  = sc.nextInt();

        for(int i=sendernode;i != recivernode; i = (i + 1) % n){
            System.out.print( i + "->");
        }
        System.out.println(recivernode);
    }
}
```

Assignment 6:

```
BullyAlgorithm.java
import java.util.*;
public class BullyAlgorithm {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter Number of Process : ");
        int n = sc.nextInt();

        int[] process = new int[n];
        System.out.println("Enter Process ID's");
        for(int i=0;i<n;i++){
            process[i] = sc.nextInt();
        }

        System.out.println("Enter Crashed Process : ");
        int crashed = sc.nextInt();

        System.out.println("Enter initiator Process : ");
        int initiator = sc.nextInt();
        System.out.println("Election Message sent to highest Proess : ");
        for(int i=0;i<n;i++){
            if(process[i] > initiator && process[i] != crashed){
                System.out.println("Process " + initiator + "-> Process " + process[i]);
            }
        }

        int newleader = -1;
        for(int i=n-1;i>=0;i--){
            if(process[i] != crashed){
                newleader = process[i];
                break;
            }
        }
        System.out.println("New Coordinator in Process : " + newleader);
    }

}
```

RingElection.java

```
import java.util.*;
public class RingElection {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter Number Process : ");
        int n = sc.nextInt();
```

```
int[] process = new int[n];
System.out.println("Enter Process ID'S : ");
for(int i =0;i<n;i++){
    process[i] = sc.nextInt();
}

System.out.println("Enter crashed Process ID : ");
int crashed = sc.nextInt();

System.out.println("Enter Initiator Process ID : ");
int initiator = sc.nextInt();

int index = 0;
for(int i=0;i<n;i++){
    if(process[i] == initiator){
        index = i;
        break;
    }
}

System.out.println("Election Message is Passesd...");

int newleader = -1;
for(int i=0;i<n;i++){
    int current = process[(index + i) % n];
    if(current != crashed){
        System.out.println(current + " - > ");
        newleader = Math.max(newleader, current);
```

```
            }
        }
        System.out.println("Back to process" + initiator);
        System.out.println("New Coordinator is process : " + newleader);
    }
}
```