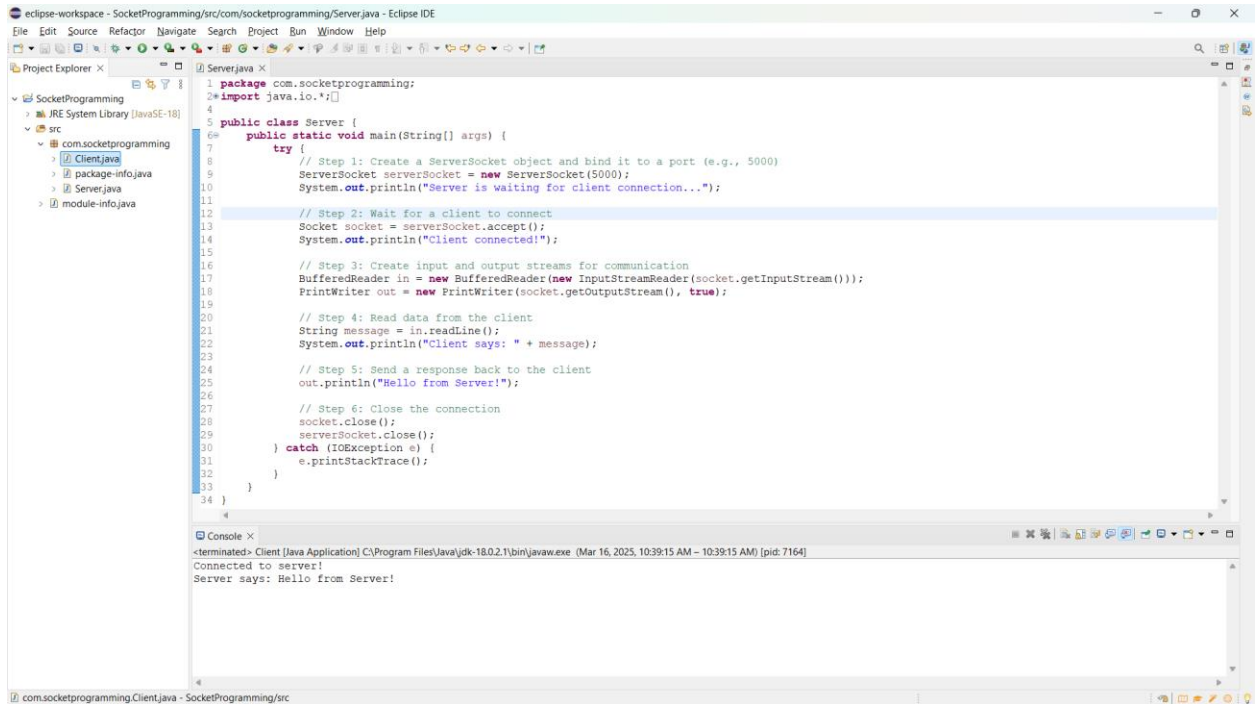


1. Socket Programming:

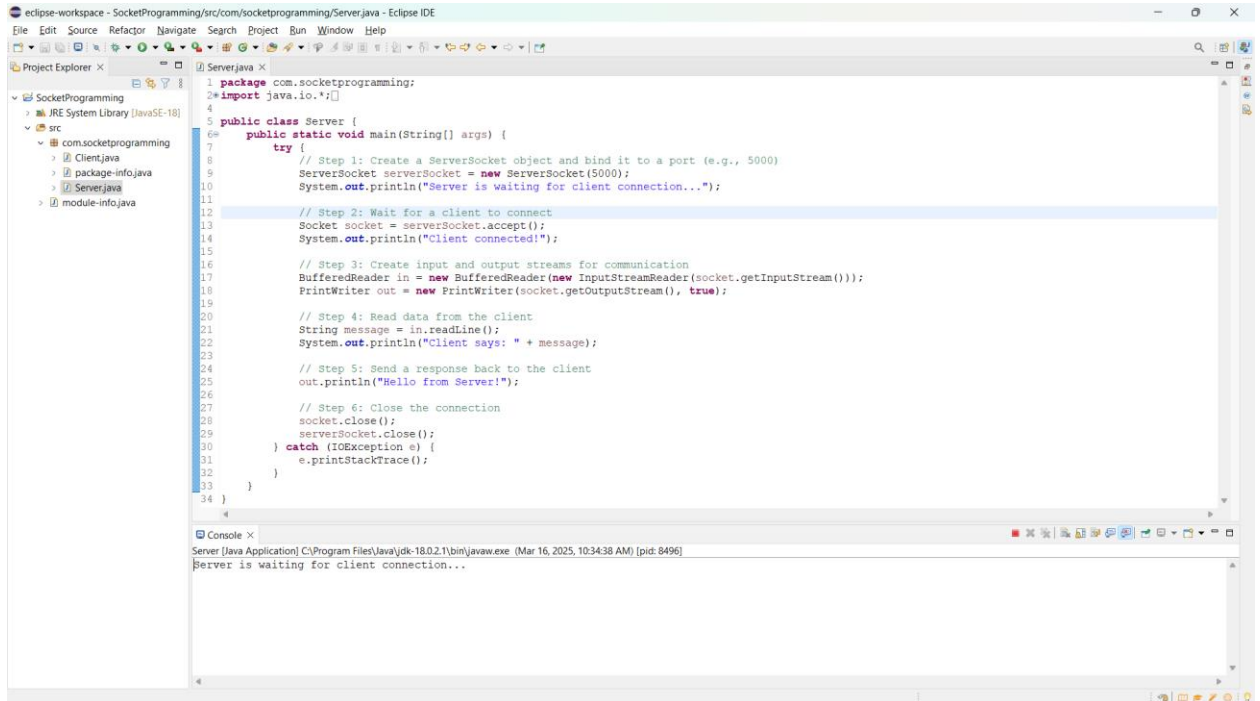


The screenshot shows the Eclipse IDE with a project named 'SocketProgramming'. The 'Server.java' file is open, displaying the following code:

```
1 package com.socketprogramming;
2 import java.io.*;
3
4
5 public class Server {
6     public static void main(String[] args) {
7         try {
8             // Step 1: Create a ServerSocket object and bind it to a port (e.g., 5000)
9             ServerSocket serverSocket = new ServerSocket(5000);
10            System.out.println("Server is waiting for client connection...");
11
12            // Step 2: Wait for a client to connect
13            Socket socket = serverSocket.accept();
14            System.out.println("Client connected!");
15
16            // Step 3: Create input and output streams for communication
17            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
18            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
19
20            // Step 4: Read data from the client
21            String message = in.readLine();
22            System.out.println("Client says: " + message);
23
24            // Step 5: Send a response back to the client
25            out.println("Hello from Server!");
26
27            // Step 6: Close the connection
28            socket.close();
29            serverSocket.close();
30        } catch (IOException e) {
31            e.printStackTrace();
32        }
33    }
34 }
```

The console output shows the following messages:

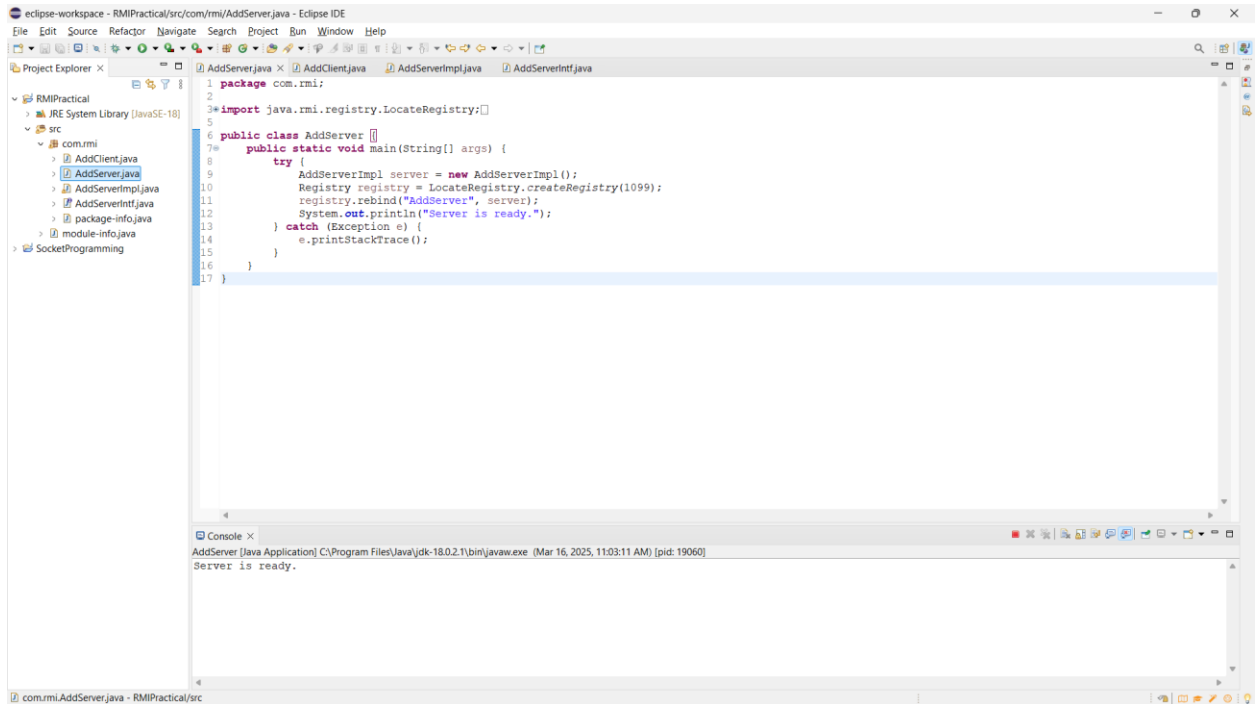
```
<terminated> Client [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (Mar 16, 2025, 10:39:15 AM - 10:39:15 AM) [pid: 7164]
Connected to server!
Server says: Hello from Server!
```



The screenshot shows the Eclipse IDE with the same 'Server.java' file open. The console output shows the following messages:

```
Server [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (Mar 16, 2025, 10:34:38 AM) [pid: 8496]
Server is waiting for client connection...
```

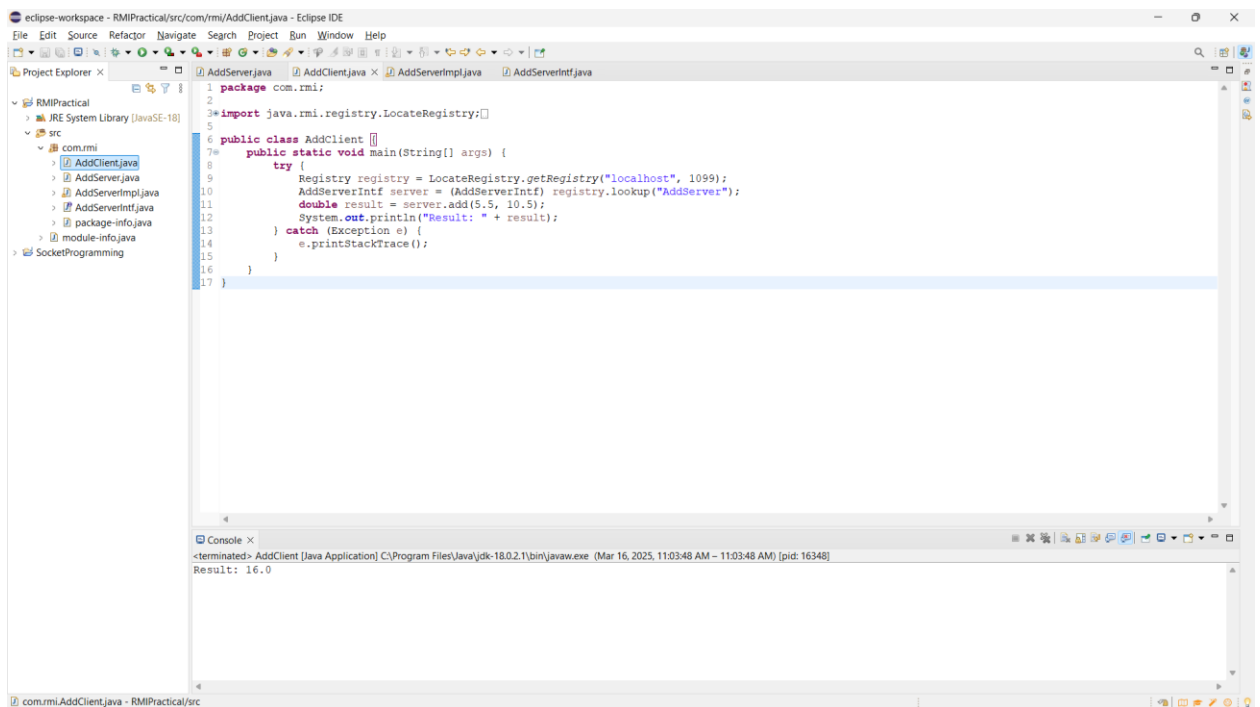
2. Remote Method Invocation (RMI):



```
1 package com.rmi;
2
3 import java.rmi.registry.LocateRegistry;
4
5
6 public class AddServer {
7     public static void main(String[] args) {
8         try {
9             AddServerImpl server = new AddServerImpl();
10            Registry registry = LocateRegistry.createRegistry(1099);
11            registry.rebind("AddServer", server);
12            System.out.println("Server is ready.");
13        } catch (Exception e) {
14            e.printStackTrace();
15        }
16    }
17 }
```

Console:

```
AddServer [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (Mar 16, 2025, 11:03:11 AM) [pid: 19060]
Server is ready.
```



```
1 package com.rmi;
2
3 import java.rmi.registry.LocateRegistry;
4
5
6 public class AddClient {
7     public static void main(String[] args) {
8         try {
9             Registry registry = LocateRegistry.getRegistry("localhost", 1099);
10            AddServerIntf server = (AddServerIntf) registry.lookup("AddServer");
11            double result = server.add(5.5, 10.5);
12            System.out.println("Result: " + result);
13        } catch (Exception e) {
14            e.printStackTrace();
15        }
16    }
17 }
```

Console:

```
<terminated> AddClient [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (Mar 16, 2025, 11:03:48 AM - 11:03:48 AM) [pid: 16348]
Result: 16.0
```

3. Common Object Request Broker Architecture (CORBA):

The screenshot shows a terminal window on the left and a text editor on the right. The terminal displays the following commands and output:

```
(base) ad@ad-OptiPlex-3010:~/Desktop/CORBA$ idlj -fall Calculator.idl
(base) ad@ad-OptiPlex-3010:~/Desktop/CORBA$ javac *.java CalculatorApp
/*.*.java
CalculatorApp/CalculatorStub.java:110: warning: IORCheckImpl is internal
proprietary API and may be removed in a future release
    com.sun.corba.se.impl.orbutil.IORCheckImpl.check(str, "Calculator
App._CalculatorStub");
    ^
Note: CalculatorApp/CalculatorPOA.java uses unchecked or unsafe operat
ions.
Note: Recompile with -Xlint:unchecked for details.
1 warning
(base) ad@ad-OptiPlex-3010:~/Desktop/CORBA$ orbd -ORBInitialPort 1050
-ORBInitialHost localhost
```

The text editor shows the `Calculator.idl` file with the following content:

```
module CalculatorApp {
    interface Calculator {
        double add(in double x, in double y);
        double subtract(in double x, in double y);
        double multiply(in double x, in double y);
        double divide(in double x, in double y);
    };
};
```

The screenshot shows a terminal window on the left and a text editor on the right. The terminal displays the following commands and output:

```
(base) ad@ad-OptiPlex-3010:~/Desktop/CORBA$ java CalculatorServer -ORB
InitialPort 1050 -ORBInitialHost localhost
CalculatorServer ready and waiting...
```

The text editor shows the `CalculatorClient.java` file with the following content:

```
import CalculatorApp.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;

public class CalculatorClient {
    public static void main(String args[]) {
        try {
            // Initialize the ORB
            ORB orb = ORB.init(args, null);

            // Get the root naming context
            org.omg.CORBA.Object objRef =
            orb.resolve_initial_references("NameService");
            NamingContextExt ncRef =
            NamingContextExtHelper.narrow(objRef);

            // Resolve the object reference in naming
            String name = "calculator";
            Calculator calculator =
            CalculatorHelper.narrow(ncRef.resolve_str(name));

            // Perform calculations
            double x = 10.5;
            double y = 2.5;

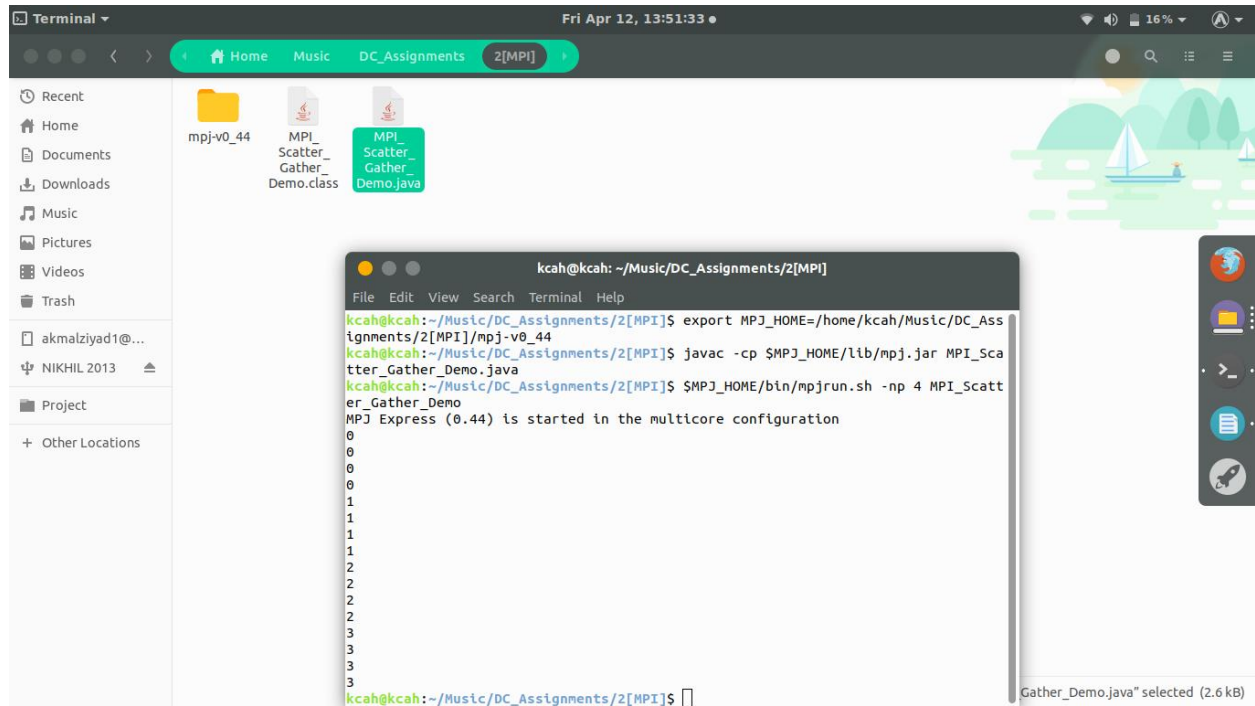
            System.out.println(x + " + " + y + " = " +
            calculator.add(x, y));
            System.out.println(x + " - " + y + " = " +
            calculator.subtract(x, y));
            System.out.println(x + " * " + y + " = " +
            calculator.multiply(x, y));
            System.out.println(x + " / " + y + " = " +
            calculator.divide(x, y));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

The screenshot shows a Linux desktop environment with a terminal window and a text editor. The terminal window, titled "ad@ad-OptiPlex-3010: ~/Desktop/CORBA", shows the execution of a Java program, CalculatorClient, which performs arithmetic operations. The text editor, titled "CalculatorServer.java", shows the source code of the CalculatorServer class, which implements a CORBA-based calculator service.

```
(base) ad@ad-OptiPlex-3010:~/Desktop/CORBA$ java CalculatorClient -ORB
InitialPort 1050 -ORBInitialHost localhost
10.5 + 2.5 = 13.0
10.5 - 2.5 = 8.0
10.5 * 2.5 = 26.25
10.5 / 2.5 = 4.2
(base) ad@ad-OptiPlex-3010:~/Desktop/CORBA$
```

```
1 import CalculatorApp.*;
2 import org.omg.CosNaming.*;
3 import org.omg.CosNaming.NamingContextPackage.*;
4 import org.omg.CORBA.*;
5 import org.omg.PortableServer.*;
6 import org.omg.PortableServer.POA;
7
8 class CalculatorImpl extends CalculatorPOA {
9     public double add(double x, double y) {
10         return x + y;
11     }
12
13     public double subtract(double x, double y) {
14         return x - y;
15     }
16
17     public double multiply(double x, double y) {
18         return x * y;
19     }
20
21     public double divide(double x, double y) {
22         if (y == 0) throw new org.omg.CORBA.BAD_PARAM("Cannot divide
23 by zero");
24         return x / y;
25 }
26
27 public class CalculatorServer {
28     public static void main(String args[]) {
29         try {
30             // Initialize the ORB
31             ORB orb = ORB.init(args, null);
32
33             // Get reference to rootpoa & activate the POAManager
34             POA rootpoa =
```

4. Message Passing Interface (MPI):

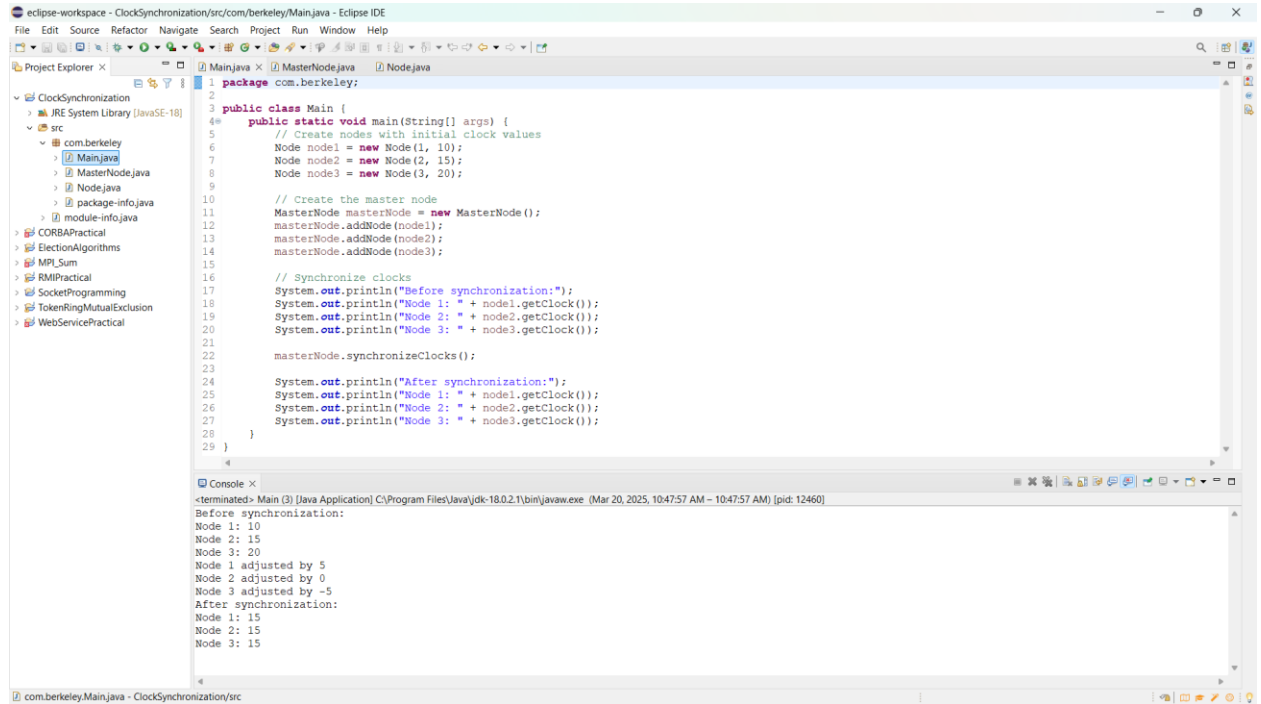


The screenshot shows a macOS desktop environment. In the background, a file manager window is open, displaying the contents of the directory `~/Music/DC_Assignments/2[MPI]`. It shows three files: `mpj-v0_44` (a folder), `MPI_Scatter_Gather_Demo.class` (a Java class file), and `MPI_Scatter_Gather_Demo.java` (a Java source file). In the foreground, a terminal window is open, showing the following commands and output:

```
kcah@kcah: ~/Music/DC_Assignments/2[MPI]
kcah@kcah:~/Music/DC_Assignments/2[MPI]$ export MPJ_HOME=/home/kcah/Music/DC_Assignments/2[MPI]/mpj-v0_44
kcah@kcah:~/Music/DC_Assignments/2[MPI]$ javac -cp $MPJ_HOME/lib/mpj.jar MPI_Scatter_Gather_Demo.java
kcah@kcah:~/Music/DC_Assignments/2[MPI]$ $MPJ_HOME/bin/mpjrun.sh -np 4 MPI_Scatter_Gather_Demo
MPJ Express (0.44) is started in the multicore configuration
0
0
0
0
1
1
1
1
1
2
2
2
2
3
3
3
3
kcah@kcah:~/Music/DC_Assignments/2[MPI]$
```

The output of the terminal shows the execution of the MPI program with 4 processes. The output consists of a sequence of numbers: 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3. A status bar at the bottom right of the terminal window indicates that the file `Gather_Demo.java` is selected (2.6 kB).

5. Clock Synchronization (Berkeley Algorithm):

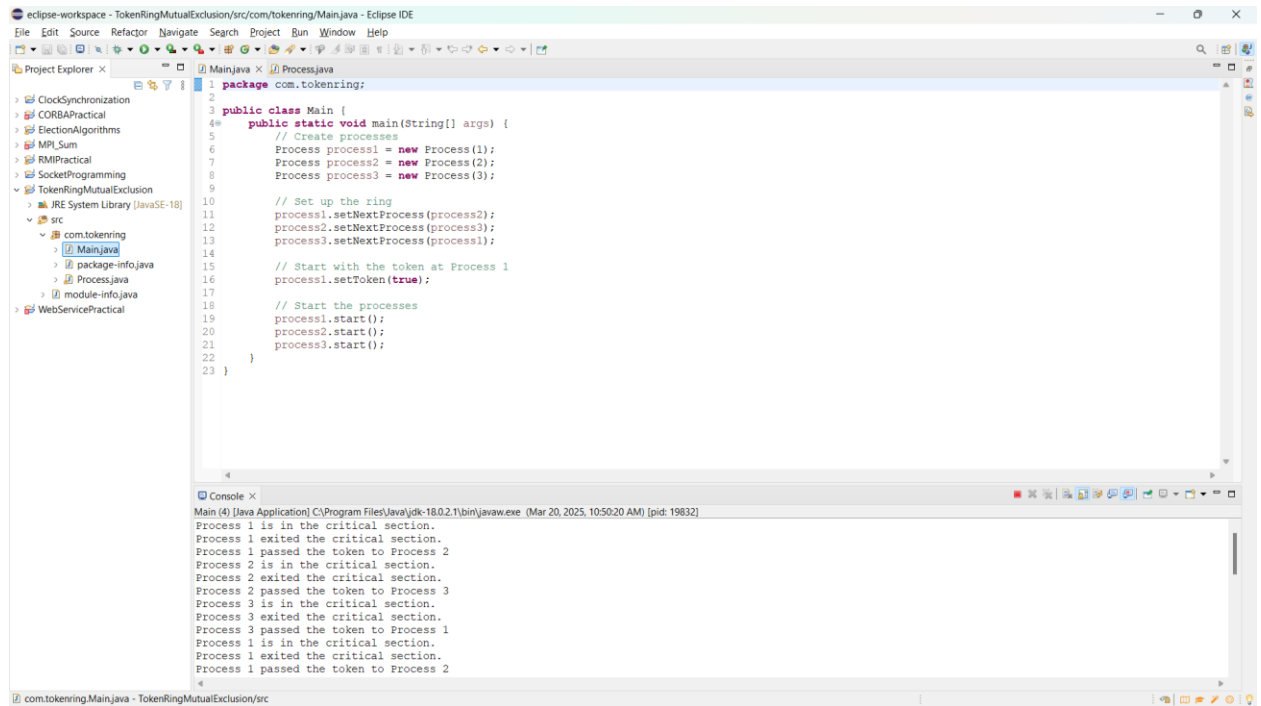


```
1 package com.berkeley;
2
3 public class Main {
4     public static void main(String[] args) {
5         // Create nodes with initial clock values
6         Node node1 = new Node(1, 10);
7         Node node2 = new Node(2, 15);
8         Node node3 = new Node(3, 20);
9
10        // Create the master node
11        MasterNode masterNode = new MasterNode();
12        masterNode.addNode(node1);
13        masterNode.addNode(node2);
14        masterNode.addNode(node3);
15
16        // Synchronize clocks
17        System.out.println("Before synchronization:");
18        System.out.println("Node 1: " + node1.getClock());
19        System.out.println("Node 2: " + node2.getClock());
20        System.out.println("Node 3: " + node3.getClock());
21
22        masterNode.synchronizeClocks();
23
24        System.out.println("After synchronization:");
25        System.out.println("Node 1: " + node1.getClock());
26        System.out.println("Node 2: " + node2.getClock());
27        System.out.println("Node 3: " + node3.getClock());
28    }
29 }
```

Console Output:

```
<terminated> Main (3) [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (Mar 20, 2025, 10:47:57 AM - 10:47:57 AM) [pid: 12460]
Before synchronization:
Node 1: 10
Node 2: 15
Node 3: 20
Node 1 adjusted by 5
Node 2 adjusted by 0
Node 3 adjusted by -5
After synchronization:
Node 1: 15
Node 2: 15
Node 3: 15
```

6. Mutual Exclusion (Token Ring Algorithm):



The screenshot displays the Eclipse IDE interface. The Project Explorer on the left shows a project named 'TokenRingMutualExclusion' with a source folder 'src' containing the package 'com.tokenring'. The package 'com.tokenring' contains three files: 'Main.java', 'Process.java', and 'module-info.java'. The 'Main.java' file is open in the editor, showing the following code:

```
1 package com.tokenring;
2
3 public class Main {
4     public static void main(String[] args) {
5         // Create processes
6         Process process1 = new Process(1);
7         Process process2 = new Process(2);
8         Process process3 = new Process(3);
9
10        // Set up the ring
11        process1.setNextProcess(process2);
12        process2.setNextProcess(process3);
13        process3.setNextProcess(process1);
14
15        // Start with the token at Process 1
16        process1.setToken(true);
17
18        // Start the processes
19        process1.start();
20        process2.start();
21        process3.start();
22    }
23 }
```

The Console window at the bottom shows the output of the program, demonstrating the token ring algorithm's execution:

```
Main (4) [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (Mar 20, 2025, 10:50:20 AM) [pid: 19832]
Process 1 is in the critical section.
Process 1 exited the critical section.
Process 1 passed the token to Process 2
Process 2 is in the critical section.
Process 2 exited the critical section.
Process 2 passed the token to Process 3
Process 3 is in the critical section.
Process 3 exited the critical section.
Process 3 passed the token to Process 1
Process 1 is in the critical section.
Process 1 exited the critical section.
Process 1 passed the token to Process 2
```

7. Election Algorithms (Bully and Ring):

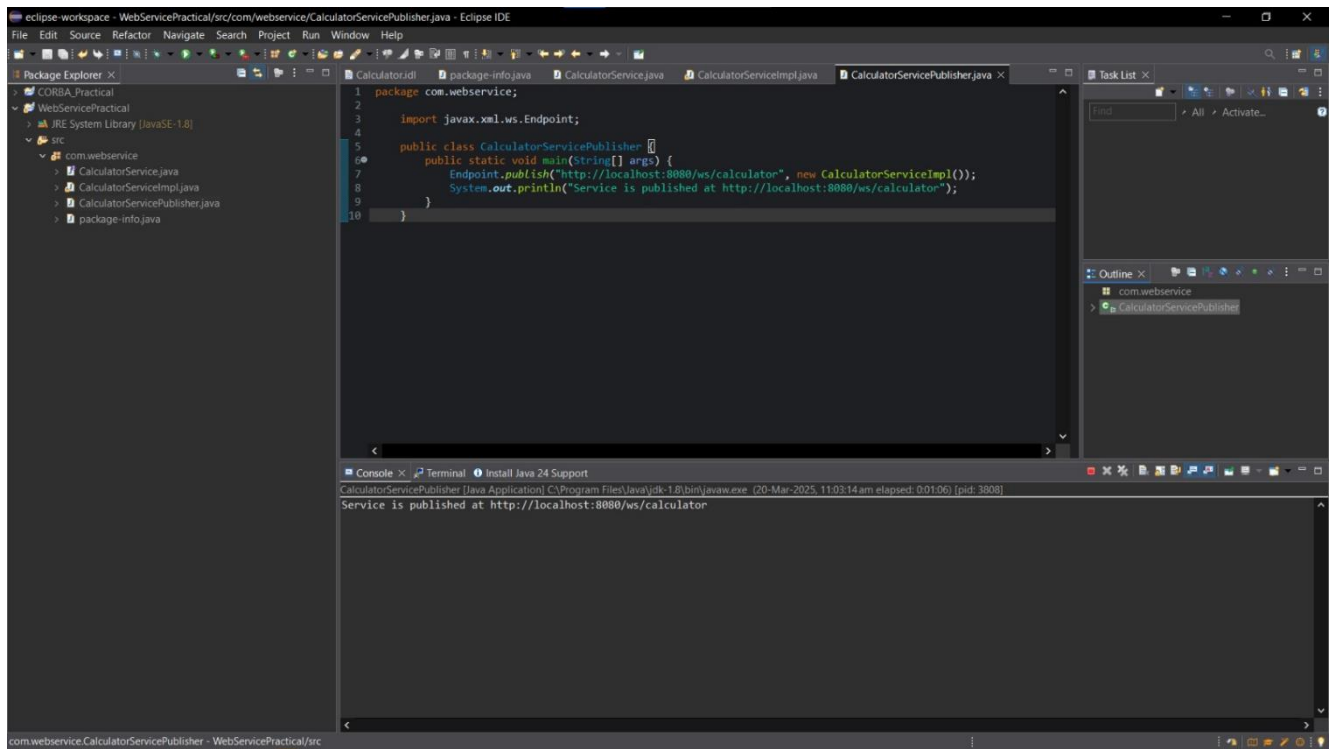
The screenshot shows the Eclipse IDE with the project 'ElectionAlgorithms' open. The 'Main.java' file is selected in the Project Explorer. The code implements the Bully Election Algorithm. It defines a package 'com.election' and a class 'Main'. The 'main' method creates three 'Process' objects (process1, process2, process3) and adds them to each other. It then starts the election from Process 1. The console output shows the following sequence of events:

```
<terminated> Main (5) [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (Mar 20, 2025, 11:12:34 AM - 11:12:35 AM) [pid: 13296]
Process 1 started an election.
Process 1 sent election message to Process 2
Process 2 responded to Process 1
Process 2 started an election.
Process 2 sent election message to Process 3
Process 3 responded to Process 2
Process 3 started an election.
Process 3 is the new coordinator.
Process 1 acknowledged Process 3 as coordinator.
Process 2 acknowledged Process 3 as coordinator.
```

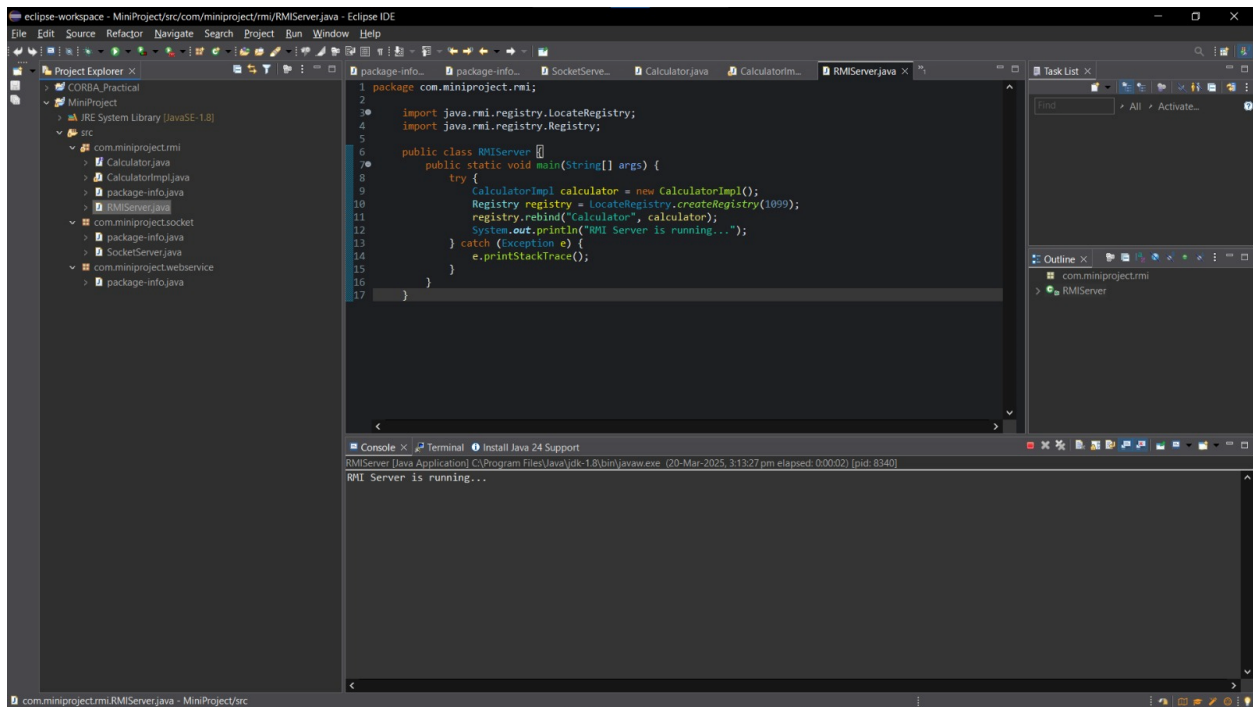
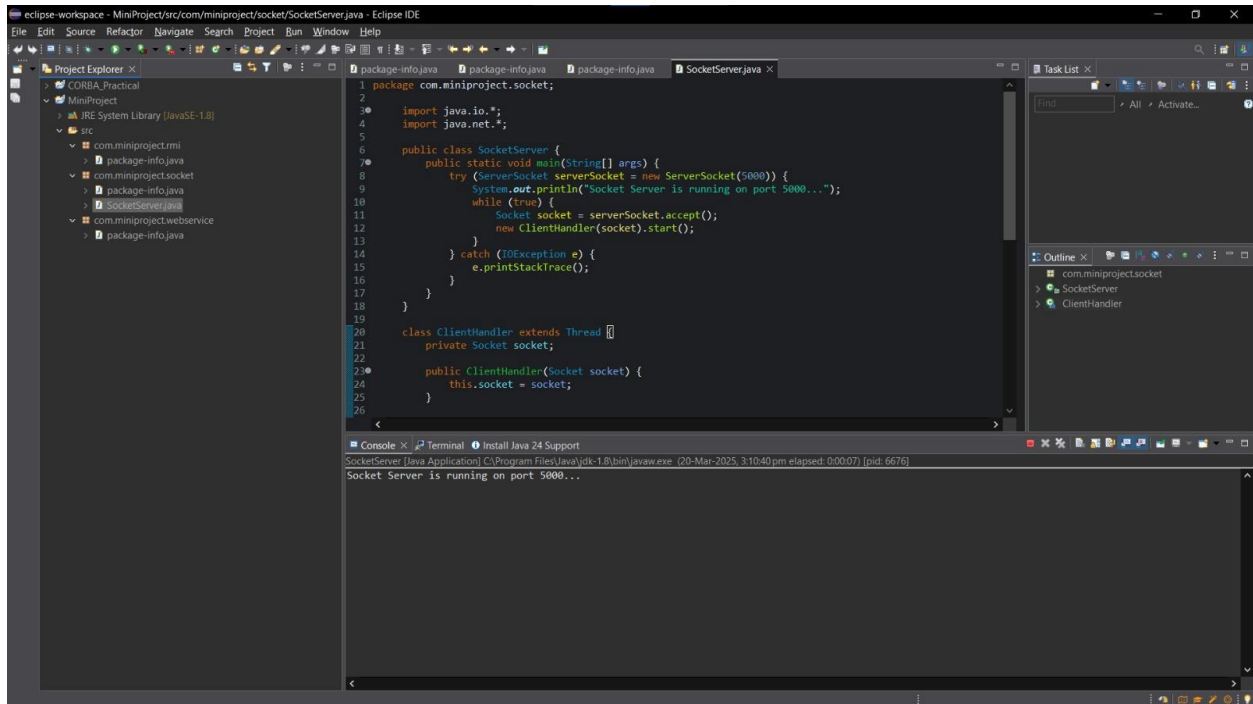
The screenshot shows the Eclipse IDE with the project 'ElectionAlgorithms' open. The 'Main.java' file is selected in the Project Explorer. The code implements the Ring Election Algorithm. It defines a package 'com.election' and a class 'Main'. The 'main' method creates three 'RingProcess' objects (process1, process2, process3) and sets up the ring by calling 'setNextProcess' on each. It then starts the election from Process 1. The console output shows the following sequence of events:

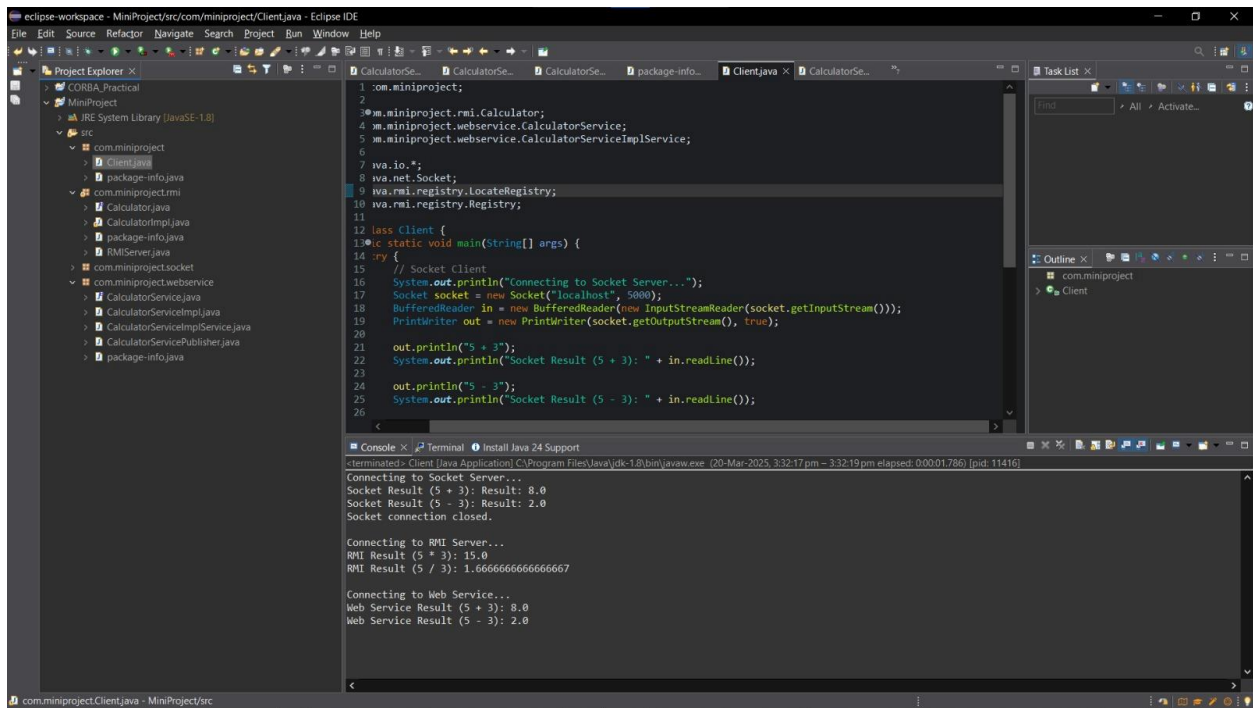
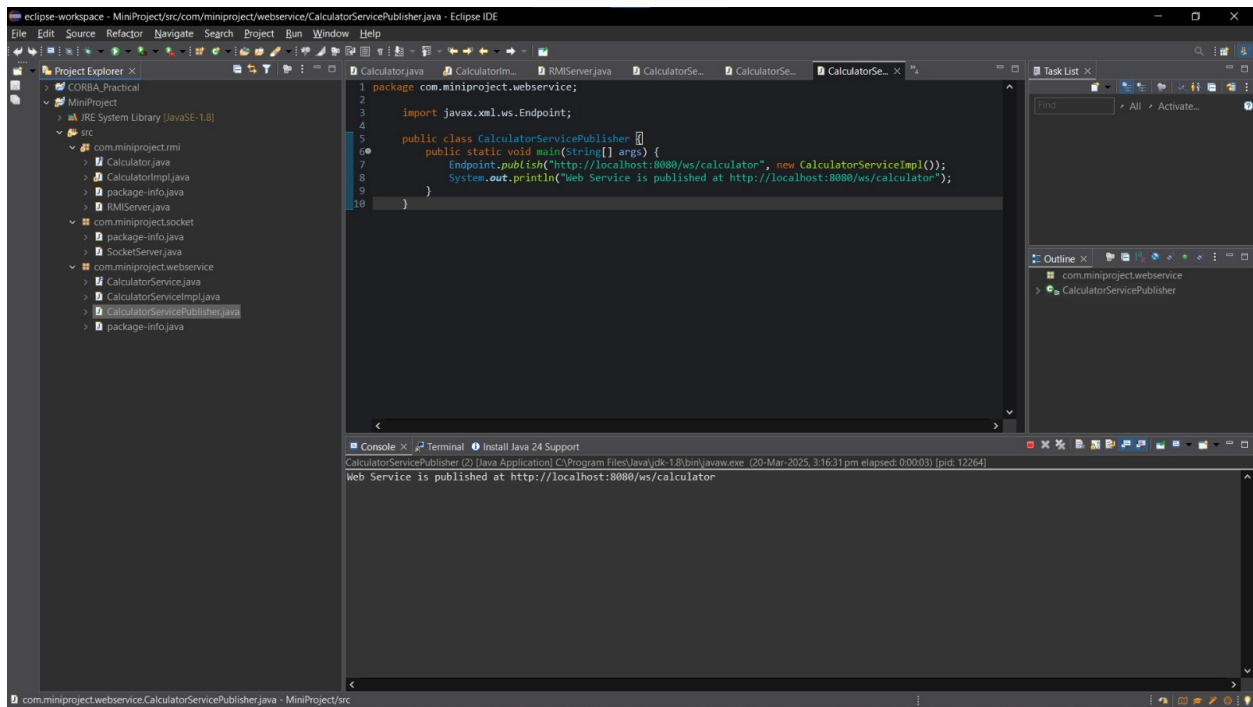
```
<terminated> Main (5) [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (Mar 20, 2025, 11:13:35 AM - 11:13:36 AM) [pid: 13988]
Process 1 started an election.
Process 3 is the new coordinator.
Process 1 acknowledged Process 3 as coordinator.
Process 2 acknowledged Process 3 as coordinator.
Process 3 acknowledged Process 3 as coordinator.
```


8. Web Services:



9. Mini Project:





S.M.