EXPERIMENT NO. 6

Program:

```
//MutualServer.java
package com.saif.exp6;

import java.io.*;
import java.net.*;
public class MutualServer implements Runnable {
   Socket socket = null;
   static ServerSocket ss;
   MutualServer(Socket newSocket) {
      this.socket = newSocket;
   }


   public static void main(String[] args) throws IOException {
      ss = new ServerSocket(7000);
      System.out.println("Server Started");
      while (true) {
         Socket s = ss.accept();
         MutualServer es = new MutualServer(s);
         Thread t = new Thread(es);
         t.start();
      }

   }

   @Override
   public void run() {
      try {
         BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
         while (true) {
            System.out.println(in.readLine());
         }

      } catch (Exception e) {}
   }

}


//ClientOne.java
package com.saif.exp6;
```

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

public class ClientOne {
   public static void main(String[] args) throws IOException {
      Socket s = new Socket("127.0.0.1", 7000);
      PrintStream out = new PrintStream(s.getOutputStream());
      ServerSocket ss = new ServerSocket(7001);
      Socket s1 = ss.accept();
      BufferedReader in1 = new BufferedReader(new InputStreamReader(s1.getInputStream()));
      PrintStream out1 = new PrintStream(s1.getOutputStream());
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      String str = "Token";
      while (true) {
         if (str.equalsIgnoreCase("Token")) {
            System.out.println("Do you want to send some data");
            System.out.println("Enter Yes or No");
            str = br.readLine();
            if (str.equalsIgnoreCase("Yes")) {
               System.out.println("Enter the data");
               str = br.readLine();
               out.println(str);
            }

            out1.println("Token");
         }

         System.out.println("Waiting for Token");
         str = in1.readLine();
      }

   }

}


//ClientTwo.java
package com.saif.exp6;

import java.io.*;
import java.net.*;
public class ClientTwo {
   public static void main(String args[]) throws IOException {
      Socket s = new Socket("127.0.0.1", 7000);
      PrintStream out = new PrintStream(s.getOutputStream());
      Socket s2 = new Socket("127.0.0.1", 7001);
      BufferedReader in2 = new BufferedReader(new InputStreamReader(s2.getInputStream()));
      PrintStream out2 = new PrintStream(s2.getOutputStream());
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      String str = "Token";
```

```
while (true) {
                System.out.println("Waiting for Token");
                str = in2.readLine();
                if (str.equalsIgnoreCase("Token")) {
                   System.out.println("Do you want to send some data");
                   System.out.println("Enter Yes or No");
                }

                str = br.readLine();
                if (str.equalsIgnoreCase("Yes")) {
                   System.out.println("Enter the data");
                   str = br.readLine();
                   out.println(str);
                }

                out2.println("Token");
            }

        }

    }
```

Output:

```
D:\Users\Saif\Desktop\BE Computer\SEM 8\Lab Manual\Distributed Computing\exp6>javac MutualServer.java ClientOne.java ClientTwo.java

D:\Users\Saif\Desktop\BE Computer\SEM 8\Lab Manual\Distributed Computing\exp6>java MutualServer
Server Started
```

```
D:\Users\Saif\Desktop\BE Computer\SEM 8\Lab Manual\Distributed Computing\exp6>java ClientOne
Do you want to send some data
Enter Yes or No
Yes
Enter the data
Distributed Computing
Waiting for Token
Do you want to send some data
Enter Yes or No
```

```
D:\Users\Saif\Desktop\BE Computer\SEM 8\Lab Manual\Distributed Computing\exp6>java ClientTwo
Waiting for Token
Do you want to send some data
Enter Yes or No
Yes
Enter the data
Parallel Computing
Waiting for Token
```

```
D:\Users\Saif\Desktop\BE Computer\SEM 8\Lab Manual\Distributed Computing\exp6>javac MutualServer.java ClientOne.java ClientTwo.java

D:\Users\Saif\Desktop\BE Computer\SEM 8\Lab Manual\Distributed Computing\exp6>java MutualServer
Server Started
Distributed Computing
Parallel Computing
```

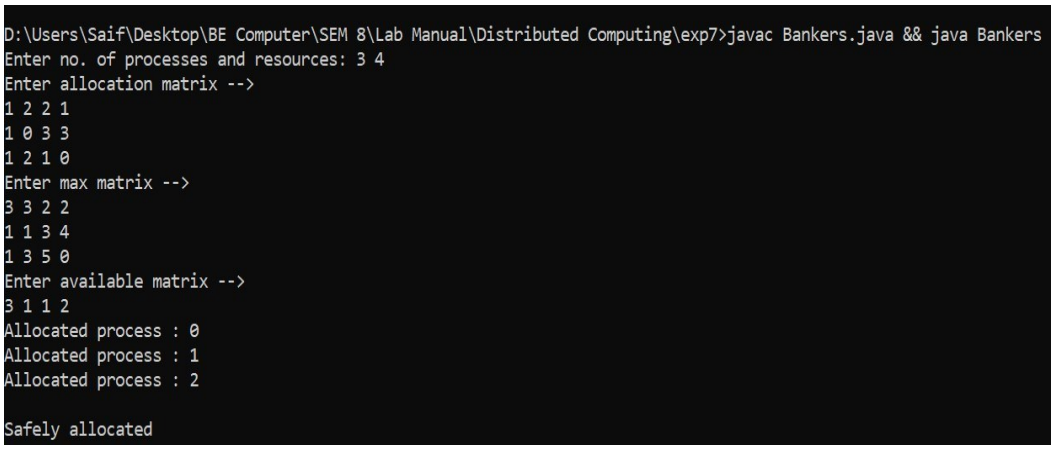
EXPERIMENT NO. 7

Program:

```
//Bankers.java
package com.saif.exp7;
Scanner sc=new Scanner(System.in);
System.out.print("Enter no. of processes and resources: ");
np=sc.nextInt();    //no. of process
nr=sc.nextInt();    //no. of resources
need=new int[np][nr];  //initializing arrays
max=new int[np][nr];
allocate= new int[np][nr];
avail=new int[1][nr];
System.out.println("Enter allocation matrix -->");
for(int i=0;i<np;i++){
   for(int j=0;j<nr;j++){
      allocate[i][j]=sc.nextInt(); //allocation
      matrix}}
System.out.println("Enter max matrix -->");
for(int i=0;i<np;i++) {
   for (int j = 0; j < nr; j++){
      max[i][j] = sc.nextInt(); //max matrix}}
System.out.println("Enter available matrix -->");
for(int j=0;j<nr;j++){
   avail[0][j]=sc.nextInt(); //available matrix}
sc.close();}
private int[][] calc_need(){
   for(int i=0;i<np;i++) {
      for (int j = 0; j < nr; j++) { //calculating need matrix
         need[i][j] = max[i][j] - allocate[i][j];}}
   return need;}
private boolean check(int i){
//checking if all resources for ith process can be allocated
   for(int j=0;j<nr;j++){
   return true;}

boolean done[]=new boolean[np]; int j=0;
   while(j<np) { //until all process allocated
      boolean allocated=false;
      for(int i=0;i<np;i++) {
         if (!done[i] && check(i)) { //trying to allocate
            for (int k = 0; k < nr; k++){
               avail[0][k]=avail[0][k]-need[i][k]+max[i][k];}
               System.out.println("Allocated process : " + i);}j++;}
      if(!allocated)  break;  //if no allocation}
   if(j==np) {  //if all processes are allocated
      System.out.println("\nSafely allocated");
   }else {
      System.out.println("All process cant be allocated safely");}}
```

```
    public static void main(String[] args){
        new Bankers().isSafe();}
```

Output:



EXPERIMENT NO. 8

Program:

```
package com.saif.exp8;

import java.util.*;

        public class LoadBalance {
           static void printLoad(int servers, int processes){
              int each = processes / servers;
              int extra = processes % servers;
              int total = 0;
              int i = 0;
              for (i = 0; i < extra; i++) {
                 System.out.println("Server "+(i+1)+" has "+(each+1)+" Processes");
              }

              for (;i<servers;i++){
                 System.out.println("Server "+(i+1)+" has "+each+" Processes");
                  }

           }

           public static void main(String[] args){
              Scanner sc = new Scanner(System.in);
              System.out.print("Enter the number of Servers: ");
              int servers= sc.nextInt();
              System.out.print("Enter the number of Processes: ");
              int processes = sc.nextInt();
              while (true){
                 printLoad(servers,processes);

                 System.out.println("\n1.Add Servers 2.Remove Server 3.Add Processes 4.Remove
        Processes 5.Exit ");
```

```
                    case 1:
                       System.out.print("How many more servers to add? ");
                       servers+=sc.nextInt();
                       System.out.print("How many more servers to remove? ");
                       servers-=sc.nextInt
                       System.out.print("How many more Processes to add? ");
                       processes+=sc.nextInt();
                       break;
        case 4:
        System.out.print("How many more processes to remove? ");
        processes-=sc.nextInt();
        break;
        case 5:
        return;}}}}
```

Output:

```
D:\Users\Saif\Desktop\BE Computer\SEM 8\Lab Manual\Distributed Computing\exp8>javac LoadBalance.java && java LoadBalance
Enter the number of Servers: 4
Enter the number of Processes: 17
Server 1 has 5 Processes
Server 2 has 4 Processes
Server 3 has 4 Processes
Server 4 has 4 Processes

1.Add Servers 2.Remove Server 3.Add Processes 4.Remove Processes 5.Exit
> 3
How many more Processes to add? 3
Server 1 has 5 Processes
Server 2 has 5 Processes
Server 3 has 5 Processes
Server 4 has 5 Processes

1.Add Servers 2.Remove Server 3.Add Processes 4.Remove Processes 5.Exit
> 4
How many more Processes to remove? 7
Server 1 has 4 Processes
Server 2 has 3 Processes
Server 3 has 3 Processes
Server 4 has 3 Processes

1.Add Servers 2.Remove Server 3.Add Processes 4.Remove Processes 5.Exit
> 1
How many more Servers to add? 1
Server 1 has 3 Processes
Server 2 has 3 Processes
Server 3 has 3 Processes
Server 4 has 2 Processes
Server 5 has 2 Processes
```

EXPERIMENT NO. 9

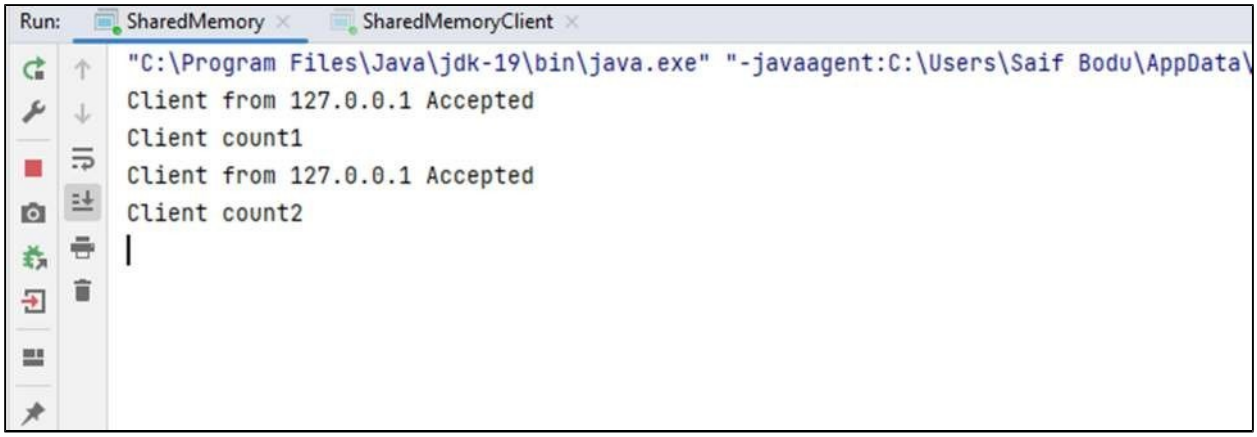
// SharedMemory.java

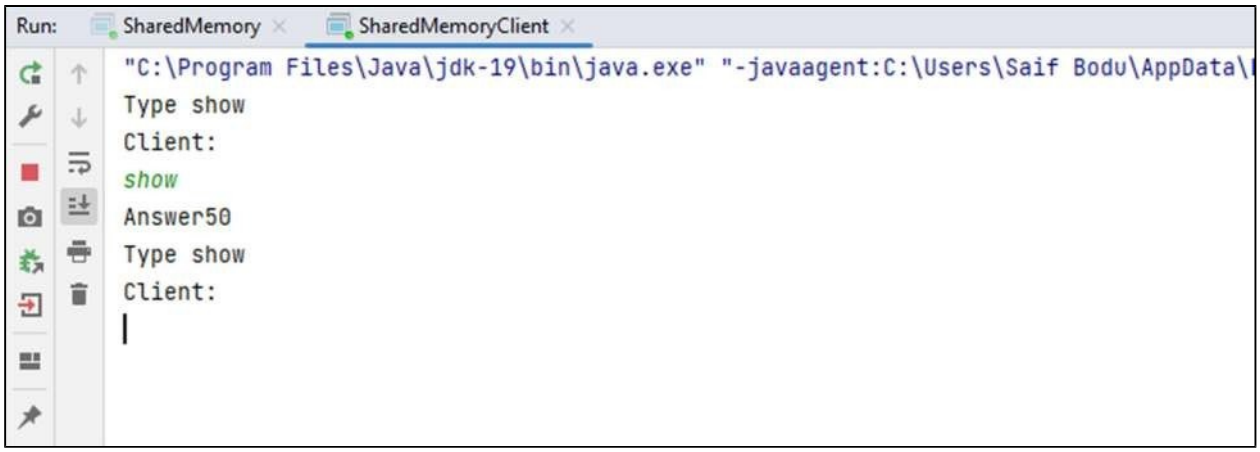
```
package com.saif.exp9;

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

public class
   SharedMemory { static
   int a = 50;
   static int count = 0;

   public static int getA(PrintStream cout) {

   public static void main(String[] args) throws
      IOException { ServerSocket ss = new
      ServerSocket(2000);
      while (true) {
         Socket sk = ss.accept();
         BufferedReader cin = new BufferedReader(new
InputStreamReader(sk.getInputStream()));
         PrintStream cout = new PrintStream(sk.getOutputStream());
         System.out.println("Client from " + sk.getInetAddress().getHostAddress() + "
Accepted");
         String s = cin.readLine();
         if (s.equalsIgnoreCase("show")) {
getA(cout);
```

// SharedMemoryClient.java

```
package com.saif.exp10;
public class SharedMemoryClient {
   public static void main(String[] args) throws
      IOException { Socket sk = new Socket("Localhost",
      2000);
      BufferedReader sin = new BufferedReader(new InputStreamReader(sk.getInputStream()));
      PrintStream sout = new PrintStream(sk.getOutputStream());
      Scanner stdin = new
      Scanner(System.in); String s;
      while (true) {
         s = sin.readLine();
         System.out.println("Answer" +
         s);}}}
```

Output:


Run:
SharedMemory
SharedMemoryClient
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Users\Saif Bodu\AppData\
Client from 127.0.0.1 Accepted
Client count1
Client from 127.0.0.1 Accepted
Client count2


Run:
SharedMemory
SharedMemoryClient
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Users\Saif Bodu\AppData\
Type show
Client:
show
Answer50
Type show
Client:

EXPERIMENT NO. 10

Program:

// DistributedFileSystemServer.java

```
package com.saif.exp10;

import

public class DistributedFileSystemServer {
   private static final int SERVER_PORT =
   12345;

   public static void main(String[] args) {
      DistributedFileSystem fileSystem = new DistributedFileSystem();
      try (ServerSocket serverSocket = new
         ServerSocket(SERVER_PORT)) { System.out.println("Server
         started on port " + SERVER_PORT);  while (true) {
            try (Socket clientSocket =
               serverSocket.accept()) {
               System.out.println("Client connected: " +
clientSocket.getInetAddress().getHostAddress());
               handleClient(clientSocket, fileSystem);}}

} catch (IOException e) {
               System.err.println("Server error: " + e.getMessage());}}
   private static void handleClient(Socket clientSocket, DistributedFileSystem fileSystem) {
      try (DataInputStream inputStream = new DataInputStream(clientSocket.getInputStream());
          DataOutputStream outputStream = new
DataOutputStream(clientSocket.getOutputStream
         ())) { while (true) {
            String command =
            inputStream.readUTF(); if
            (command.equals("add")) {
               String fileName = inputStream.readUTF();
               InetAddress address =
               InetAddress.getByName(inputStream.readUTF());
               fileSystem.addFile(fileName, address);
               outputStream.writeBoolean(true);
            } else if (command.equals("remove")) {


               String fileName = inputStream.readUTF();
               InetAddress address =
               InetAddress.getByName(inputStream.readUTF());
               fileSystem.removeFile(fileName, address);
               outputStream.writeBoolean(true);
            } else if (command.equals("get")) {
               String fileName = inputStream.readUTF();
               List<InetAddress> addresses = fileSystem.getFileAddresses(fileName);
               outputStream.writeInt(addresses.size());
               for (InetAddress address : addresses) {
                  outputStream.writeUTF(address.getHostAddress());
               }
```

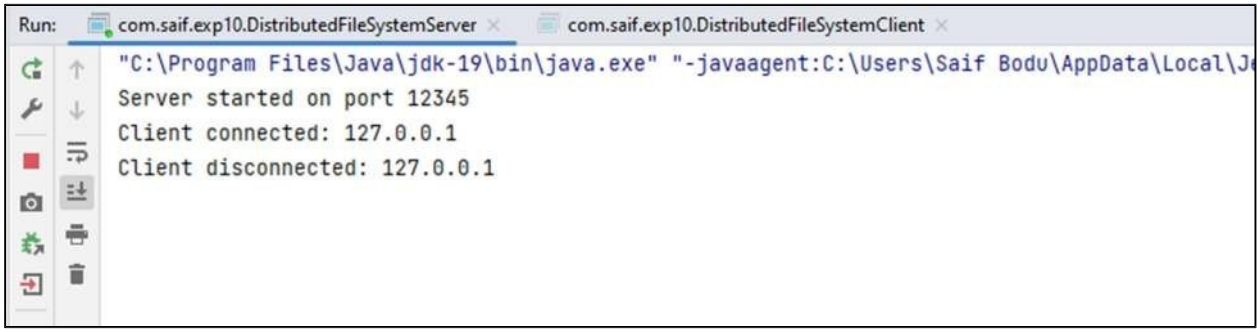
```
            } else if
               (command.equals("exit")) {
               break;
            } else {
               System.err.println("Invalid command: " + command);
               outputStream.writeBoolean(false);
            }

         }

      } catch (IOException e) {
         System.err.println("Client error: " + e.getMessage());
      }

      System.out.println("Client disconnected: " +
clientSocket.getInetAddress().getHostAddress());
   }

}
```

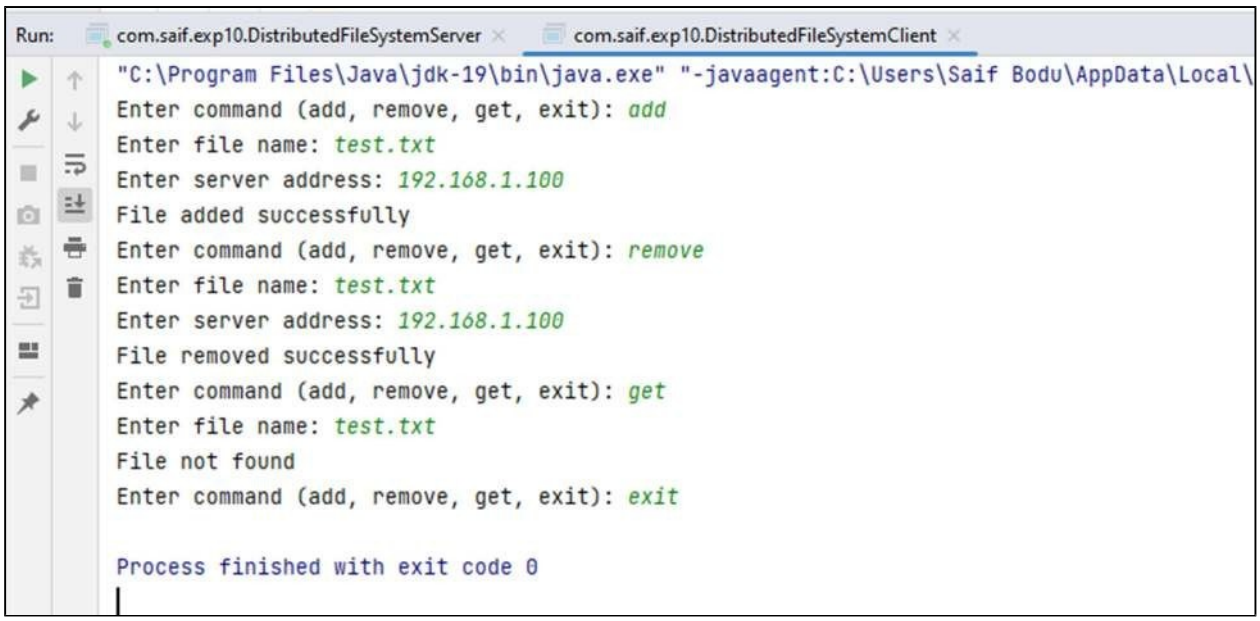
// DistributedFileSystemClient.java

```
public class DistributedFileSystemClient {
   private static final String SERVER_HOST = "localhost";
   private static final int SERVER_PORT = 12345;


   public static void main(String[] args) {
      try (Scanner scanner = new Scanner(System.in);
           Socket socket = new Socket(SERVER_HOST, SERVER_PORT);
           DataInputStream inputStream = new
           DataInputStream(socket.getInputStream());
           DataOutputStream outputStream = new
          DataOutputStream(socket.getOutputStream())) { while (true) {
            System.out.print("Enter command (add, remove, get, exit):
            "); String command = scanner.nextLine();
            if (command.equals("add")) {
               System.out.print("Enter file name: ");
               String fileName = scanner.nextLine();
               System.out.print("Enter server
               address: ");
               InetAddress address =
               InetAddress.getByName(scanner.nextLine());
               outputStream.writeUTF(command);
               outputStream.writeUTF(fileName);
               outputStream.writeUTF(address.getHostAddress());
               boolean success =
               inputStream.readBoolean(); if (success) {
                  System.out.println("File added successfully");
               } else {
                  System.out.println("Failed to add file");
               }

            } else if (command.equals("remove")) {
               System.out.print("Enter file name: ");
               String fileName = scanner.nextLine();
               System.out.print("Enter server
```

```
            address: ");
            InetAddress address =
            InetAddress.getByName(scanner.nextLine());
            outputStream.writeUTF(command);
            outputStream.writeUTF(fileName);
            outputStream.writeUTF(address.getHostAddress());
            boolean success =
            inputStream.readBoolean(); if (success) {
                System.out.println("File removed successfully");
            } else {
                System.out.println("Failed to remove file");
            }

        } else if (command.equals("get")) {
            System.out.print("Enter file name:
            ");
            if (numAddresses == 0) {
                System.out.println("File not found");
            } else {
                System.out.println("File found on " + numAddresses + "
                servers:"); for (int i = 0; i < numAddresses; i++) {
                    String address =
                    inputStream.readUTF();
                    System.out.println("- " + address);
                }

            }

        } else if (command.equals("exit")) {
            outputStream.writeUTF(command);
            break;
```

Output:




Run:
com.saif.exp10.DistributedFileSystemServer
com.saif.exp10.DistributedFileSystemClient
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Users\Saif Bodu\AppData\Local\
Enter command (add, remove, get, exit): add
Enter file name: test.txt
Enter server address: 192.168.1.100
File added successfully
Enter command (add, remove, get, exit): remove
Enter file name: test.txt
Enter server address: 192.168.1.100
File removed successfully
Enter command (add, remove, get, exit): get
Enter file name: test.txt
File not found
Enter command (add, remove, get, exit): exit
Process finished with exit code 0