Practical No. - 11

Aim: Demonstration of various reader and writer subclasses in listing

Program:

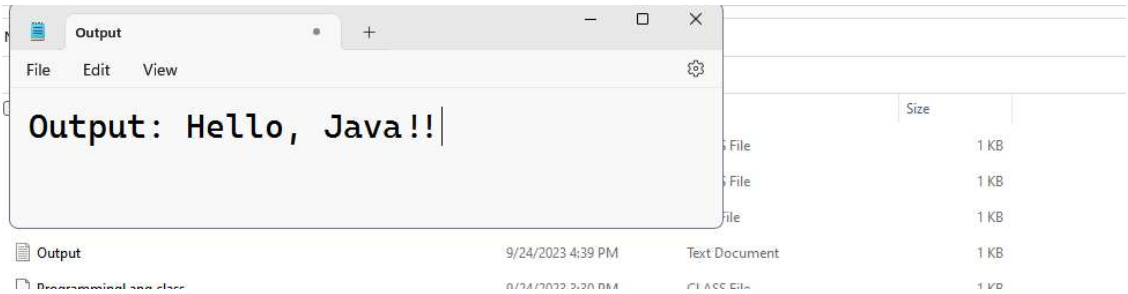
```
import java.io.*;

public class ReaderWriter {
    public static void main(String args[]) throws IOException {
        System.out.println("With InputStreamReader");
        int a;
        String s;

        // Create an InputStreamReader to read input from the console
        InputStreamReader inr = new InputStreamReader(System.in);
        System.out.print("Enter a line: ");

        // Read characters until Enter (ASCII 13) is pressed
        while ((a = inr.read()) != 13) {
            System.out.print((char) a);
        }

        System.out.println(); // Print a newline
        System.out.println("\nWith BufferedReader and
        InputStreamReader");

        // Create a BufferedReader to read input from the console using
        InputStreamReader
        BufferedReader br = new BufferedReader(new
        InputStreamReader(System.in));
        System.out.print("Enter a line: ");

        // Read a line of text from the console
        String inputLine = br.readLine();
        System.out.println("You entered: " + inputLine);

        System.out.println("\nOutput With PrintWriter and
        FileWriter");

        // Create a BufferedReader to read input from the console
        using InputStreamReader
        BufferedReader br1 = new BufferedReader(new
        InputStreamReader(System.in));

        // Create a PrintWriter to write output to a file named
        "Output.txt"
        PrintWriter p = new PrintWriter(new FileWriter("Output.txt"));

        System.out.print("Enter lines (Ctrl+C to exit): ");

        // Read lines from the console and write them to the file with a
        prefix
```

```
        while ((s = br1.readLine()) != null) {
            p.println("Output: " + s);
        }

        // Close the PrintWriter to save the changes to the file
        p.close();
    }
}
```

Output:

```
With InputStreamReader
Enter a line: Hello, Java!!
Hello, Java!!

With BufferedReader and InputStreamReader
Enter a line: Hello, Java!!
You entered: Hello, Java!!

Output With PrintWriter and FileWriter
Enter lines (Ctrl+C to exit): Hello, Java!!
```

Output on file:



Practical No. - 12

Aim : Write a java program using runnable interface and with the help of thread class , create three threads. Run each thread 3 times and then stop thread excution.

Program:

```
// Define a class A that implements the Runnable interface
class A implements Runnable {
    public void run() {
        int i;
        for (i = 1; i <= 3; i++) {
            System.out.println("Thread A : " + i);
        }
    }
}

// Define a class B that implements the Runnable interface
class B implements Runnable {
    public void run() {
        int i;
        for (i = 1; i <= 3; i++) {
            System.out.println("Thread B : " + i);
        }
    }
}

// Define a class C that implements the Runnable interface
class C implements Runnable {
    public void run() {
        int i;
        for (i = 1; i <= 3; i++) {
            System.out.println("Thread C : " + i);
        }
    }
}

// Define a class RunnableDemo
class RunnableDemo {
    public static void main(String hello[]) throws Exception {
        System.out.println("Main starts");

        // Create three thread objects (t1, t2, t3) associated with A,
        B, and C
        Thread t1 = new Thread(new A());
        Thread t2 = new Thread(new B());
        Thread t3 = new Thread(new C());

        // Start the threads
        t1.start();
        t2.start();
```

```
        t3.start();

        // Wait for all threads to finish
        t1.join();
        t2.join();
        t3.join();

        System.out.println("Main ends");
    }// end main()
}// end class
```

Output:

```
Main starts
Thread C : 1
Thread C : 2
Thread A : 1
Thread B : 1
Thread B : 2
Thread C : 3
Thread A : 2
Thread A : 3
Thread B : 3
Main ends
```

Practical No. - 13

Aim: Write a program to create 4 threads to perform 4 different arithmetic operations like addition, subtraction, multiplication and division. Accept two numbers from command line arguments and perform the operations using thread.

Program:

```
import java.util.Scanner;

// Create a class called 'Add' that extends the Thread class
class Add extends Thread {
    int n1, n2;

    // Constructor for the 'Add' class
    public Add(int x, int y) {
        n1 = x;
        n2 = y;
    }

    // Override the 'run' method to perform addition and print the
    result
    @Override
    public void run() {
        System.out.println("Addition is : " + (n1 + n2));
    }
}

// Create a class called 'Sub' that extends the Thread class
class Sub extends Thread {
    int n1, n2;

    // Constructor for the 'Sub' class
    public Sub(int x, int y) {
        n1 = x;
        n2 = y;
    }

    // Override the 'run' method to perform subtraction and print the
    result
    @Override
    public void run() {
        System.out.println("Subtraction is : " + (n1 - n2));
    }
}

// Create a class called 'Mul' that extends the Thread class
class Mul extends Thread {
    int n1, n2;

    // Constructor for the 'Mul' class
```

```
    public Mul(int x, int y) {
        n1 = x;
        n2 = y;
    }

    // Override the 'run' method to perform multiplication and print
    the result
    @Override
    public void run() {
        System.out.println("Multiplication is : " + (n1 * n2));
    }
}

// Create a class called 'Div' that extends the Thread class
class Div extends Thread {
    int n1, n2;

    // Constructor for the 'Div' class
    public Div(int x, int y) {
        n1 = x;
        n2 = y;
    }

    // Override the 'run' method to perform division and print the
    result or handle division by zero
    @Override
    public void run() {
        if (n2 != 0) {
            System.out.println("Division is : " + (n1 / n2));
        } else {
            System.out.println("Division by zero is not allowed.");
        }
    }
}

// Create a class called 'ThreadDemo' to demonstrate the usage of the above
thread classes
class ThreadDemo {
    public static void main(String ar[]) {
        try {
            Scanner scanner = new Scanner(System.in);

            // Prompt the user to enter two numbers
            System.out.print("Enter 1st number: ");
            int a = scanner.nextInt();
            System.out.print("Enter 2nd number: ");
            int b = scanner.nextInt();

            // Create instances of the thread classes and start them
            new Add(a, b).start();
            new Sub(a, b).start();
            new Mul(a, b).start();
            new Div(a, b).start();
        } catch (Exception e) {
```

```
            // Handle exceptions that may occur during user input or thread
execution
            System.err.println("An error occurred: " + e.getMessage());
        }
    }
}
```

Output:

```
Enter 1st number: 4
Enter 2nd number: 5
Subtraction is : -1
Division is : 0
Addition is : 9
Multiplication is : 20
```

Practical No. - 14

Aim: Write a client socket that will accept n names from user and send them to the server. After receiving the names , the server socket should send the message “names received: and close the connection.

Program:

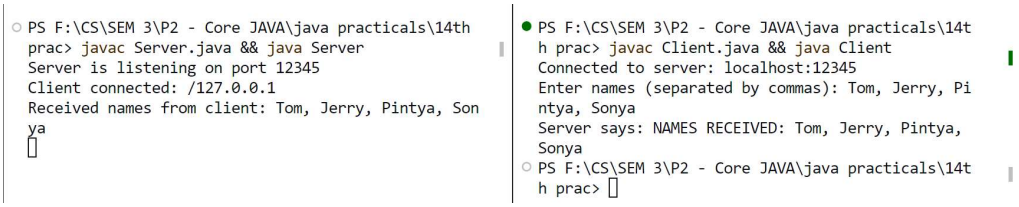
Server code:

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

public class Server {
    public static void main(String[] args) {
        final int port = 12345; // Specify the port you want to use
        try {
            ServerSocket serverSocket = new ServerSocket(port);
            System.out.println("Server is listening on port " + port);

            while (true) {
                // Wait for a client to connect
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected: " +
                clientSocket.getInetAddress());

                // Create a BufferedReader to read data from the
                client
                BufferedReader reader = new BufferedReader(new
                InputStreamReader(clientSocket.getInputStream()));
                // Create a PrintWriter to send data to the client
                PrintWriter writer = new
                PrintWriter(clientSocket.getOutputStream(), true);

                // Read a line of text (names) sent by the client
                String receivedNames = reader.readLine();
                System.out.println("Received names from client: " +
                receivedNames);

                // Send a confirmation message back to the client
                writer.println("NAMES RECEIVED: " + receivedNames);

                // Close the writer, reader, and the client socket
                writer.close();
                reader.close();
                clientSocket.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

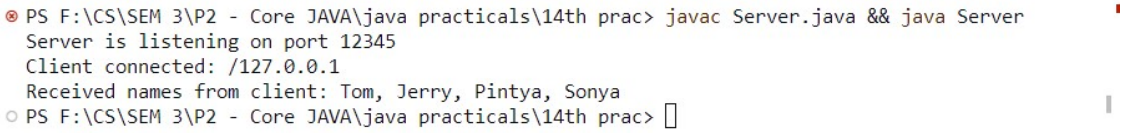
Client code:

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

public class Client {
    public static void main(String[] args) {
        final String serverAddress = "localhost"; // Change to the
        server's IP if needed
        final int serverPort = 12345; // Specify the server's port

        try {
            // Create a socket and connect to the server at the
            specified address and port
            Socket socket = new Socket(serverAddress, serverPort);
            System.out.println("Connected to server: " + serverAddress
            + ":" + serverPort);

            // Create a BufferedReader to read data from the server
            BufferedReader reader = new BufferedReader(new
            InputStreamReader(socket.getInputStream()));
            // Create a PrintWriter to send data to the server
            PrintWriter writer = new
            PrintWriter(socket.getOutputStream(), true);
            // Create a Scanner to read input from the user
            Scanner scanner = new Scanner(System.in);

            // Prompt the user to enter names separated by commas
            System.out.print("Enter names (separated by commas): ");
            String names = scanner.nextLine();

            // Send the names to the server
            writer.println(names);

            // Receive and display the server's confirmation message
            String confirmationMessage = reader.readLine();
            System.out.println("Server says: " + confirmationMessage);

            // Close the socket, reader, and writer
            socket.close();
            reader.close();
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

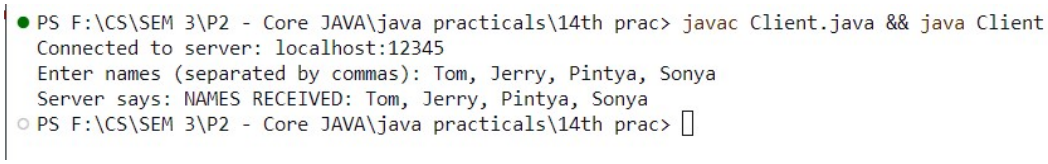
Output:



Server Output:



Client Output:



Practical No. - 15

Aim: Create a client socket which sends a number to the server. The server socket returns the sum of digits of the number if the number is positive, otherwise it sends an error message and close the connection.

Program:
Server code:

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

public class Server {
    public static void main(String[] args) {
        final int port = 12345; // Specify the port you want to use
        try {
            ServerSocket serverSocket = new ServerSocket(port);
            System.out.println("Server is listening on port " + port);

            while (true) {
                // Wait for a client to connect
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected: " +
                clientSocket.getInetAddress());

                // Create a BufferedReader to read data from the
                client
                BufferedReader reader = new BufferedReader(new
                InputStreamReader(clientSocket.getInputStream()));
                // Create a PrintWriter to send data to the client
                PrintWriter writer = new
                PrintWriter(clientSocket.getOutputStream(), true);

                // Read a line of text sent by the client
                String clientInput = reader.readLine();

                try {
                    // Attempt to parse the client input as an integer
                    int number = Integer.parseInt(clientInput);
                    System.out.println("Received number from client: "
                    + number);

                    if (number >= 0) {
                        // Calculate the sum of digits if the number
                        is non-negative
                        int sumOfDigits =
                        calculateSumOfDigits(number);
                        writer.println("Sum of digits: " +
                        sumOfDigits);
                    } else {
                        // Send an error message if the number is
```

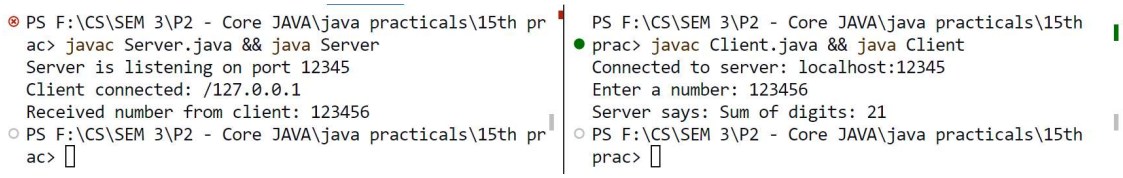
```
                    negative
                    writer.println("Error: Negative number not
                    allowed");
                }
            } catch (NumberFormatException e) {
                // Send an error message for invalid input
                writer.println("Error: Invalid input");
            }

            // Close the writer, reader, and the client socket
            writer.close();
            reader.close();
            clientSocket.close();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

// Helper method to calculate the sum of digits in a number
private static int calculateSumOfDigits(int number) {
    int sum = 0;
    while (number != 0) {
        sum += number % 10;
        number /= 10;
    }
    return sum;
}
}
```

Client code:

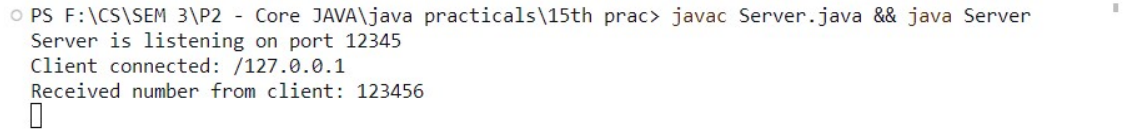
```
import java.io.*;
import java.net.*;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) {
        final String serverAddress = "localhost"; // Change to the
        server's IP if needed
        final int serverPort = 12345; // Specify the server's port

        try {
            // Create a socket and connect to the server at the
            specified address and port
            Socket socket = new Socket(serverAddress, serverPort);
            System.out.println("Connected to server: " + serverAddress
            + ":" + serverPort);

            // Create a BufferedReader to read data from the server
            BufferedReader reader = new BufferedReader(new
            InputStreamReader(socket.getInputStream()));
            // Create a PrintWriter to send data to the server
```

```
            PrintWriter writer = new
            PrintWriter(socket.getOutputStream(), true);
            // Create a Scanner to read input from the user
            Scanner scanner = new Scanner(System.in);

            // Prompt the user to enter a number
            System.out.print("Enter a number: ");
            String input = scanner.nextLine();

            // Send the entered number to the server
            writer.println(input);

            // Receive and display the server's response
            String serverResponse = reader.readLine();
            System.out.println("Server says: " + serverResponse);

            // Close the socket, reader, and writer
            socket.close();
            reader.close();
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:



Server output:



Client output: