Practical no. -11

Aim: Demonstration of various reader and writer subclasses in listing

Program:

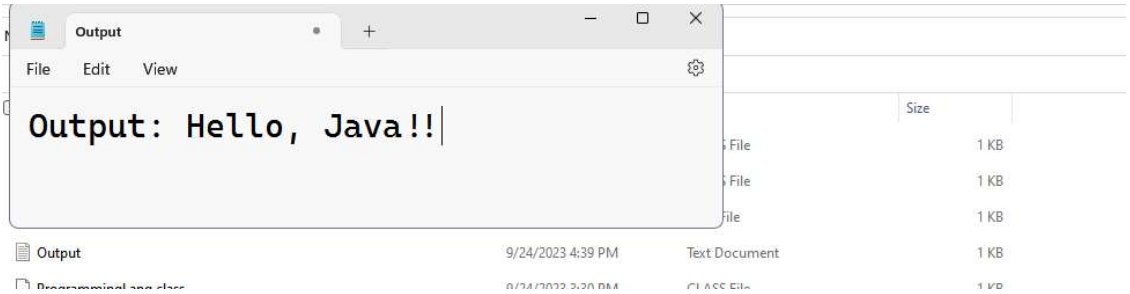
```
import java.io.*;

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

        InputStreamReader inr = new InputStreamReader(System.in);
        System.out.print("Enter a line: ");

        while ((a = inr.read()) != 13) {
            System.out.print((char) a);
        }

        System.out.println();
        System.out.println("\nWith BufferedReader and InputStreamReader");

        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        System.out.print("Enter a line: ");

        String inputLine = br.readLine();
        System.out.println("You entered: " + inputLine);

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

        BufferedReader br1 = new BufferedReader(new
InputStreamReader(System.in));
        PrintWriter p = new PrintWriter(new FileWriter("Output.txt"));

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

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

        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 10 times and then stop thread excution.

Program:

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

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

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

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

        Thread t1 = new Thread(new A());
        Thread t2 = new Thread(new B());
        Thread t3 = new Thread(new C());

        t1.start();
        t2.start();
        t3.start();

        t1.join();
        t2.join();
        t3.join();

        System.out.println("Main ends");
```

```
    }
}
```

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;

class Add extends Thread {
    int n1, n2;

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

    @Override
    public void run() {
        System.out.println("Addition is : " + (n1 + n2));
    }
}

class Sub extends Thread {
    int n1, n2;

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

    @Override
    public void run() {
        System.out.println("Subtraction is : " + (n1 - n2));
    }
}

class Mul extends Thread {
    int n1, n2;

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

    @Override
    public void run() {
        System.out.println("Multiplication is : " + (n1 * n2));
    }
}
```

```
class Div extends Thread {
    int n1, n2;

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

    @Override
    public void run() {
        if (n2 != 0) {
            System.out.println("Division is : " + (n1 / n2));
        } else {
            System.out.println("Division by zero is not allowed.");
        }
    }
}

class ThreadDemo {
    public static void main(String ar[]) {
        try {
            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter 1st number: ");
            int a = scanner.nextInt();
            System.out.print("Enter 2nd number: ");
            int b = scanner.nextInt();

            new Add(a, b).start();
            new Sub(a, b).start();
            new Mul(a, b).start();
            new Div(a, b).start();
        } catch (Exception e) {
            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:

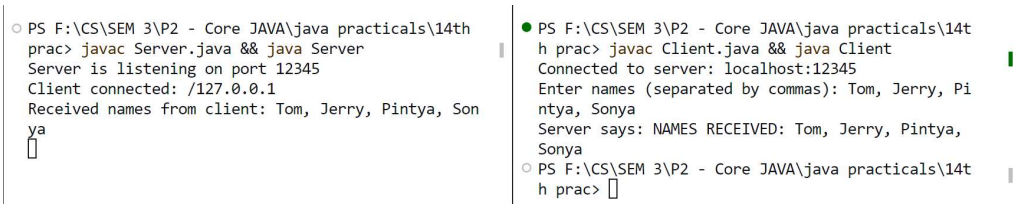
```
import java.io.*;
import java.net.*;

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

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected: " +
clientSocket.getInetAddress());

                BufferedReader reader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                PrintWriter writer = new
PrintWriter(clientSocket.getOutputStream(), true);

                String receivedNames = reader.readLine();
                System.out.println("Received names from client: " +
receivedNames);

                writer.println("NAMES RECEIVED: " + receivedNames);

                writer.close();
                reader.close();
                clientSocket.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Client code:

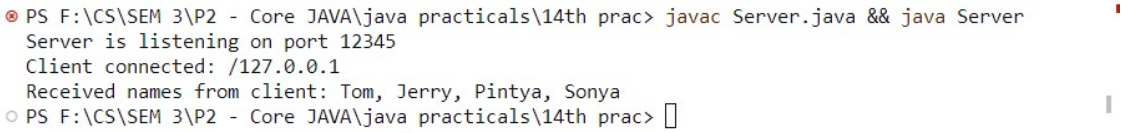
```
import java.io.*;
import java.net.*;
import java.util.Scanner;

public class Client {
```

```
    public static void main(String[] args) {
        final String serverAddress = "localhost";
        final int serverPort = 12345;

        try {
            Socket socket = new Socket(serverAddress, serverPort);
            System.out.println("Connected to server: " + serverAddress + ":"
+ serverPort);

            BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter writer = new PrintWriter(socket.getOutputStream(),
true);
            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter names (separated by commas): ");
            String names = scanner.nextLine();

            writer.println(names);

            String confirmationMessage = reader.readLine();
            System.out.println("Server says: " + confirmationMessage);

            socket.close();
            reader.close();
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

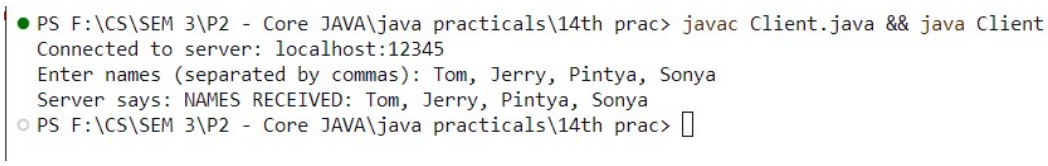
Output:



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:

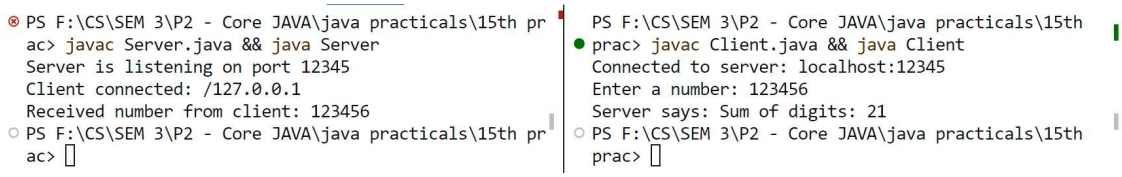
```
import java.io.*;
import java.net.*;

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

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected: " +
clientSocket.getInetAddress());

                BufferedReader reader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                PrintWriter writer = new
PrintWriter(clientSocket.getOutputStream(), true);

                String clientInput = reader.readLine();
                try {
                    int number = Integer.parseInt(clientInput);
                    System.out.println("Received number from client: " +
number);

                    if (number >= 0) {
                        int sumOfDigits = calculateSumOfDigits(number);
                        writer.println("Sum of digits: " + sumOfDigits);
                    } else {
                        writer.println("Error: Negative number not allowed");
                    }
                } catch (NumberFormatException e) {
                    writer.println("Error: Invalid input");
                }

                writer.close();
                reader.close();
                clientSocket.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
    }

    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";
        final int serverPort = 12345;

        try {
            Socket socket = new Socket(serverAddress, serverPort);
            System.out.println("Connected to server: " + serverAddress + ":"
+ serverPort);

            BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter writer = new PrintWriter(socket.getOutputStream(),
true);
            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter a number: ");
            String input = scanner.nextLine();
            writer.println(input);

            String serverResponse = reader.readLine();
            System.out.println("Server says: " + serverResponse);

            socket.close();
            reader.close();
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:



Server output:

```
PS F:\CS\SEM 3\P2 - Core JAVA\java practicals\15th prac> javac Server.java && java Server
Server is listening on port 12345
Client connected: /127.0.0.1
Received number from client: 123456
```

Client output:

```
PS F:\CS\SEM 3\P2 - Core JAVA\java practicals\15th prac> javac Client.java && java Client
Connected to server: localhost:12345
Enter a number: 123456
Server says: Sum of digits: 21
PS F:\CS\SEM 3\P2 - Core JAVA\java practicals\15th prac>
```