**CNSL Lab Assignment 8**

**Neeti Kurulkar**

**Write a program using TCP socket for wired network**

1. **Server Program**

**import** java**.**io**.\*;**

**import** java**.**net**.\*;**

**import** java**.**util**.**Stack**;**

public class Server **{**

private static final int PORT **=** 8080**;**

private static final int BUFFER\_SIZE **=** 1024**;**

// Function to evaluate a mathematical expression using JavaScript engine

public static double evaluateExpression**(**String expression**)** **{**

// Using a custom method to evaluate the expression

**return** evaluate**(**expression**);**

**}**

// Utility method to evaluate an arithmetic expression

private static double evaluate**(**String expression**)** **{**

char**[]** tokens **=** expression**.**toCharArray**();**

// Stack for numbers

Stack**<**Double**>** values **=** **new** Stack**<>();**

// Stack for operators

Stack**<**Character**>** ops **=** **new** Stack**<>();**

**for** **(**int i **=** 0**;** i **<** tokens**.**length**;** i**++)** **{**

// Current token is a whitespace, skip it

**if** **(**tokens**[**i**]** **==** ' '**)**

**continue;**

// Current token is a number, push it to stack for numbers

**if** **(**tokens**[**i**]** **>=** '0' **&&** tokens**[**i**]** **<=** '9'**)** **{**

StringBuilder buffer **=** **new** StringBuilder**();**

// There may be more than one digit in the number

**while** **(**i **<** tokens**.**length **&&** **(**tokens**[**i**]** **>=** '0' **&&** tokens**[**i**]** **<=** '9' **||** tokens**[**i**]** **==** '.'**))** **{**

buffer**.**append**(**tokens**[**i**++]);**

**}**

values**.**push**(**Double**.**parseDouble**(**buffer**.**toString**()));**

// Correct the i index after inner loop

i**--;**

**}**

// Current token is an operator

**else** **if** **(**tokens**[**i**]** **==** '+' **||** tokens**[**i**]** **==** '-' **||** tokens**[**i**]** **==** '\*' **||** tokens**[**i**]** **==** '/'**)** **{**

**while** **(!**ops**.**isEmpty**()** **&&** hasPrecedence**(**tokens**[**i**],** ops**.**peek**()))** **{**

values**.**push**(**applyOp**(**ops**.**pop**(),** values**.**pop**(),** values**.**pop**()));**

**}**

ops**.**push**(**tokens**[**i**]);**

**}**

**}**

// Apply remaining operators to remaining values

**while** **(!**ops**.**isEmpty**())** **{**

values**.**push**(**applyOp**(**ops**.**pop**(),** values**.**pop**(),** values**.**pop**()));**

**}**

// Top of 'values' contains result

**return** values**.**pop**();**

**}**

// Returns true if 'op2' has higher or same precedence as 'op1'

private static boolean hasPrecedence**(**char op1**,** char op2**)** **{**

**if** **(**op2 **==** '(' **||** op2 **==** ')'**)**

**return** **false;**

**if** **((**op1 **==** '\*' **||** op1 **==** '/'**)** **&&** **(**op2 **==** '+' **||** op2 **==** '-'**))**

**return** **false;**

**else**

**return** **true;**

**}**

// A method to apply an operator 'op' on operands 'a' and 'b'

private static double applyOp**(**char op**,** double b**,** double a**)** **{**

**switch** **(**op**)** **{**

**case** '+'**:**

**return** a **+** b**;**

**case** '-'**:**

**return** a **-** b**;**

**case** '\*'**:**

**return** a **\*** b**;**

**case** '/'**:**

**if** **(**b **==** 0**)**

**throw** **new** UnsupportedOperationException**(**"Cannot divide by zero"**);**

**return** a **/** b**;**

**}**

**return** 0**;**

**}**

// Function to handle communication with the client

public static void handleClient**(**Socket clientSocket**)** **{**

**try** **{**

BufferedReader in **=** **new** BufferedReader**(new** InputStreamReader**(**clientSocket**.**getInputStream**()));**

PrintWriter out **=** **new** PrintWriter**(**clientSocket**.**getOutputStream**(),** **true);**

// Send a greeting message to the client

out**.**println**(**"Server establishment successful"**);**

String clientMessage **=** in**.**readLine**();**

System**.**out**.**println**(**"Client says: " **+** clientMessage**);**

// File transfer

String fileName **=** in**.**readLine**();**

System**.**out**.**println**(**"Client requested file: " **+** fileName**);**

File file **=** **new** File**(**fileName**);**

**if** **(!**file**.**exists**())** **{**

out**.**println**(**"File not found."**);**

**}** **else** **{**

out**.**println**(**"File found, sending now..."**);**

**try** **(**BufferedReader fileReader **=** **new** BufferedReader**(new** FileReader**(**file**)))** **{**

String fileContent**;**

**while** **((**fileContent **=** fileReader**.**readLine**())** **!=** **null)** **{**

out**.**println**(**fileContent**);**

**}**

**}**

**}**

// Calculator logic

**while** **(true)** **{**

String expression **=** in**.**readLine**();**

**if** **(**expression**.**equalsIgnoreCase**(**"exit"**))** **{**

**break;**

**}**

// Evaluate the expression

double result **=** evaluateExpression**(**expression**);**

out**.**println**(**"Result: " **+** result**);**

**}**

// Close the socket after conversation

clientSocket**.**close**();**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"Error handling client: " **+** e**.**getMessage**());**

**}**

**}**

public static void main**(**String**[]** args**)** **{**

**try** **(**ServerSocket serverSocket **=** **new** ServerSocket**(**PORT**))** **{**

System**.**out**.**println**(**"Server is listening on port " **+** PORT**);**

**while** **(true)** **{**

Socket clientSocket **=** serverSocket**.**accept**();**

System**.**out**.**println**(**"Connection established with a client."**);**

handleClient**(**clientSocket**);**

**}**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"Server error: " **+** e**.**getMessage**());**

**}**

**}**

**}**

1. **Client Program**

**import** java**.**io**.\*;**

**import** java**.**net**.\*;**

**import** java**.**util**.**Scanner**;**

public class Client **{**

private static final String SERVER\_IP **=** "127.0.0.1"**;** // Replace with the correct server IP

private static final int PORT **=** 8080**;**

private static final int BUFFER\_SIZE **=** 1024**;**

public static void main**(**String**[]** args**)** **{**

Socket socket **=** **null;**

**try** **{**

// Create socket and connect to the server

socket **=** **new** Socket**(**SERVER\_IP**,** PORT**);**

System**.**out**.**println**(**"Connected to server at " **+** SERVER\_IP **+** ":" **+** PORT**);**

// Create input/output streams for communication

BufferedReader in **=** **new** BufferedReader**(new** InputStreamReader**(**socket**.**getInputStream**()));**

PrintWriter out **=** **new** PrintWriter**(**socket**.**getOutputStream**(),** **true);**

Scanner scanner **=** **new** Scanner**(**System**.**in**);**

char**[]** buffer **=** **new** char**[**BUFFER\_SIZE**];**

// Receive greeting from server

in**.**read**(**buffer**,** 0**,** BUFFER\_SIZE**);**

System**.**out**.**println**(**"Server says: " **+** **new** String**(**buffer**).**trim**());**

// Send greeting to server

out**.**println**(**"Hello from client"**);**

// Request a file from the server

System**.**out**.**print**(**"Enter the file name to request: "**);**

String fileName **=** scanner**.**nextLine**();**

out**.**println**(**fileName**);**

// Receive server's response for file request

in**.**read**(**buffer**,** 0**,** BUFFER\_SIZE**);**

String serverResponse **=** **new** String**(**buffer**).**trim**();**

System**.**out**.**println**(**"Server response: " **+** serverResponse**);**

// If file is found, receive the content

**if** **(**serverResponse**.**equals**(**"File found, sending now..."**))** **{**

in**.**read**(**buffer**,** 0**,** BUFFER\_SIZE**);**

System**.**out**.**println**(**"File content: " **+** **new** String**(**buffer**).**trim**());**

**}**

// Calculator functionality

**while** **(true)** **{**

char**[]** result **=** **new** char**[**BUFFER\_SIZE**];**

System**.**out**.**print**(**"Enter a mathematical expression (or type 'exit' to quit): "**);**

String expression **=** scanner**.**nextLine**();**

// Send the expression to the server

out**.**println**(**expression**);**

// Exit if the user typed 'exit'

**if** **(**expression**.**equalsIgnoreCase**(**"exit"**))** **{**

**break;**

**}**

// Receive the result from the server

in**.**read**(**result**,** 0**,** BUFFER\_SIZE**);**

System**.**out**.**println**(**"Server response: " **+** **new** String**(**result**).**trim**());**

**}**

**}** **catch** **(**IOException e**)** **{**

System**.**err**.**println**(**"Connection error: " **+** e**.**getMessage**());**

**}** **finally** **{**

// Close the socket connection

**try** **{**

**if** **(**socket **!=** **null)** **{**

socket**.**close**();**

**}**

**}** **catch** **(**IOException e**)** **{**

System**.**err**.**println**(**"Failed to close the socket: " **+** e**.**getMessage**());**

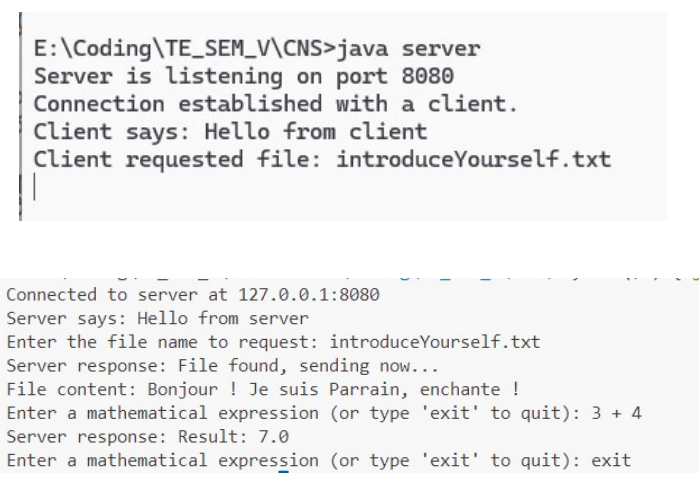
**}**

**}**

**}**

**}**

1. **Output**

****