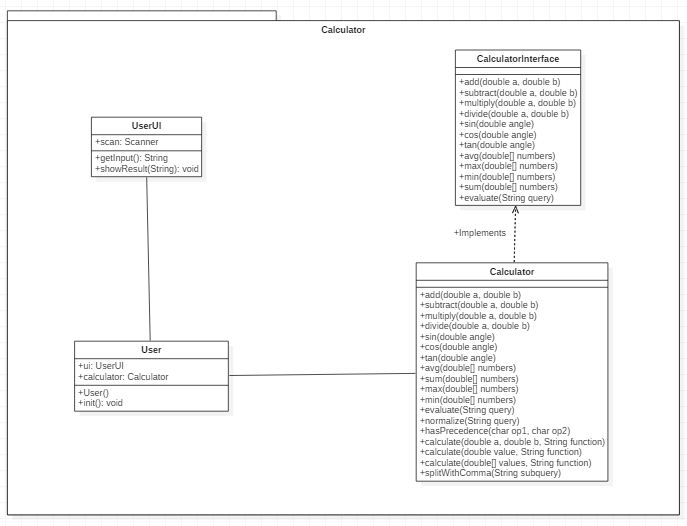
**Tutorial 1**

**Problem statement:**

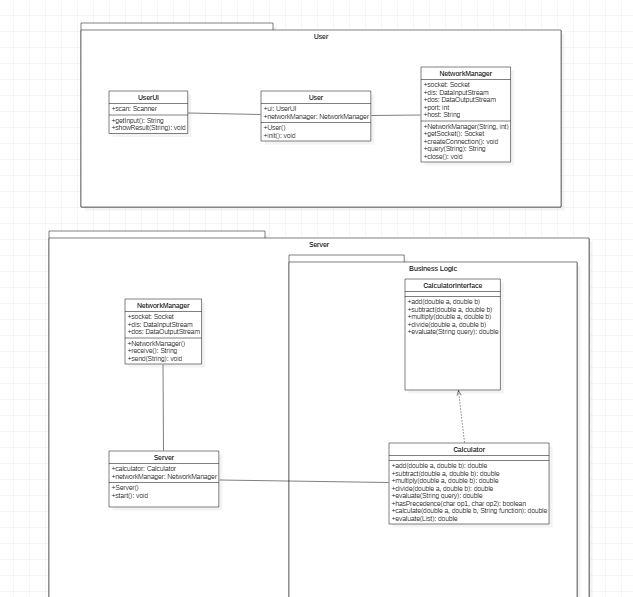
**Design Assumptions:**

**Design Diagrams:**

**Monolithic architecture:**

****

**Client server architecture:**

****

**Code:**

**Calculator monolithic :**

**//calc class**

package com.mangnaik.yogesh.calculator;

import java.util.ArrayList;

import java.util.List;

import java.util.Stack;

public class Calculator implements CalculatorInterface {

@Override

public double add(double a, double b) {

return a+b;

}

@Override

public double subtract(double a, double b) {

return a-b;

}

@Override

public double multiply(double a, double b) {

return a\*b;

}

@Override

public double divide(double a, double b) {

return a/b;

}

@Override

public double evaluate(String query) {

query = query.replaceAll("pi", "3.1415926535");

List<String> list = new ArrayList<>();

//query = normalize(query);

convertToPostFix(list, query);

return evaluate(list);

}

private double evaluate(List<String> list){

Stack<String> stack = new Stack<>();

for(int i=0; i<list.size(); i++){

if(list.get(i).equals("+")||list.get(i).equals("-")||list.get(i).equals("\*")||list.get(i).equals("/")){

double a = Double.parseDouble(stack.pop());

double b = Double.parseDouble(stack.pop());

stack.push(""+calculate(a,b,list.get(i).charAt(0)+""));

}

else{

stack.push(list.get(i));

}

}

return Double.valueOf(stack.pop());

}

private void convertToPostFix(List<String> list, String query) {

Stack<Character> operators = new Stack<>();

char[] tokens = query.toCharArray();

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

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

StringBuilder stringBuilder = new StringBuilder();

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

stringBuilder.append(tokens[i++]);

}

i--;

list.add(stringBuilder.toString());

}

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

operators.push('(');

else if (tokens[i] == ')'){

while (operators.peek() != '(')

list.add(operators.pop()+"");

operators.pop();

}

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

while (!operators.empty() && hasPrecedence(tokens[i], operators.peek())){

list.add(operators.pop()+"");

}

operators.push(tokens[i]);

}

}

while (!operators.empty())

list.add(operators.pop()+"");

}

private boolean hasPrecedence(char op1, char op2) {

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

return false;

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

return false;

else

return true;

}

//calling arithmetic functions

private double calculate(double a, double b, String function){

switch (function){

case "+":

return add(a,b);

case "-":

return subtract(b,a);

case "\*":

return multiply(a,b);

case "/":

return divide(b,a);

}

return 0;

}

}

**Calculator.java (interface)**

|  |  |
| --- | --- |
| package com.mangnaik.yogesh.calculator;  /\*\*  \* Created by Yogesh on 2/8/2018.  \*/  public interface CalculatorInterface {  //basic  public double add(double a, double b);  public double subtract(double a, double b);  public double multiply(double a, double b);  public double divide(double a, double b);  public double evaluate(String query);  } | |
|  | | |
|  | | |  |
|  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |

**User.java**

package com.mangnaik.yogesh.calculator;

import java.io.IOException;

public class User {

UserUI ui;

Calculator calculator;

public static void main(String args[]) {

new User();

}

public User() {

ui = new UserUI();

calculator = new Calculator();

init();

}

private void init(){

String query = "";

while(!query.equals("exit")){

query = ui.getInput();

if(!query.equals("")){

double answer = calculator.evaluate(query);

ui.showResult(answer+"");

}

}

}

}

**Userui.java**

package com.mangnaik.yogesh.calculator;

import java.util.Scanner;

/\*\*

\* Created by Yogesh on 2/15/2018.

\*/

public class UserUI {

Scanner scan = new Scanner(System.in);

public String getInput(){

String query = scan.nextLine();

return query;

}

public void showResult(String answer){

System.out.println(answer);

}

}

**Client Server Architecture**

**Network Manger**

package com.mangnaik.yogesh.user;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.Socket;

public class NetworkManager {

Socket socket = null;

DataInputStream din = null;

DataOutputStream dout = null;

int port;

String host;

public NetworkManager(String host, int port){

this.port = port;

this.host = host;

}

private Socket getSocket() throws IOException {

return new Socket(host, port);

}

public void createConnection(){

try {

socket = getSocket();

} catch (IOException e) {

e.printStackTrace();

System.out.println ("Failed to Create Socket");

return;

}

try {

din = new DataInputStream(socket.getInputStream());

dout = new DataOutputStream(socket.getOutputStream());

} catch (IOException e) {

e.printStackTrace();

System.out.println("Failed to connect to the server");

}

}

public String query(String query) throws IOException {

dout.writeUTF(query);

String ans = din.readUTF();

return ans;

}

public void close() {

try {

socket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

**User class**

package com.mangnaik.yogesh.user;

import java.io.IOException;

/\*\*

\* Created by Yogesh on 2/2/2018.

\*/

public class User {

UserUI ui;

NetworkManager networkManager;

public static void main(String args[]) {

new User();

}

public User() {

ui = new UserUI();

init();

}

private void init(){

networkManager = new NetworkManager("localhost", 8192);

networkManager.createConnection();

String query = "";

while(!query.equals("exit")){

query = ui.getInput();

System.out.println("Query : " + query);

if(!query.equals("")){

String ans;

try {

ans = networkManager.query(query);

} catch (IOException e) {

System.out.println("Connection Reset");

networkManager.close();

return;

}

ui.showResult(ans);

}

}

}

}

**UserUI class**

package com.mangnaik.yogesh.user;

import java.util.Scanner;

/\*\*

\* Created by Yogesh on 2/15/2018.

\*/

public class UserUI {

Scanner scan = new Scanner(System.in);

public String getInput(){

String query = scan.nextLine();

return query;

}

public void showResult(String answer){

System.out.println(answer);

}

}

**Calculator**

package com.mangnaik.yogesh.server;

import java.util.ArrayList;

import java.util.List;

import java.util.Stack;

/\*\*

\* Created by Yogesh on 2/8/2018.

\*/

public class Calculator implements CalculatorInterface {

@Override

public double add(double a, double b) {

return a+b;

}

@Override

public double subtract(double a, double b) {

return a-b;

}

@Override

public double multiply(double a, double b) {

return a\*b;

}

@Override

public double divide(double a, double b) {

return a/b;

}

@Override

public double evaluate(String query) {

query = query.replaceAll("pi", "3.1415926535");

List<String> list = new ArrayList<>();

//query = normalize(query);

convertToPostFix(list, query);

return evaluate(list);

}

private double evaluate(List<String> list){

Stack<String> stack = new Stack<>();

for(int i=0; i<list.size(); i++){

if(list.get(i).equals("+")||list.get(i).equals("-")||list.get(i).equals("\*")||list.get(i).equals("/")){

double a = Double.parseDouble(stack.pop());

double b = Double.parseDouble(stack.pop());

stack.push(""+calculate(a,b,list.get(i).charAt(0)+""));

}

else{

stack.push(list.get(i));

}

}

return Double.valueOf(stack.pop());

}

private void convertToPostFix(List<String> list, String query) {

Stack<Character> operators = new Stack<>();

char[] tokens = query.toCharArray();

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

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

StringBuilder stringBuilder = new StringBuilder();

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

stringBuilder.append(tokens[i++]);

}

i--;

list.add(stringBuilder.toString());

}

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

operators.push('(');

else if (tokens[i] == ')'){

while (operators.peek() != '(')

list.add(operators.pop()+"");

operators.pop();

}

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

while (!operators.empty() && hasPrecedence(tokens[i], operators.peek())){

list.add(operators.pop()+"");

}

operators.push(tokens[i]);

}

}

while (!operators.empty())

list.add(operators.pop()+"");

}

private boolean hasPrecedence(char op1, char op2) {

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

return false;

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

return false;

else

return true;

}

//calling arithmetic functions

private double calculate(double a, double b, String function){

switch (function){

case "+":

return add(a,b);

case "-":

return subtract(b,a);

case "\*":

return multiply(a,b);

case "/":

return divide(b,a);

}

return 0;

}

}

**CalculatorInterface**

package com.mangnaik.yogesh.server;

/\*\*

\* Created by Yogesh on 2/8/2018.

\*/

public interface CalculatorInterface {

//basic

public double add(double a, double b);

public double subtract(double a, double b);

public double multiply(double a, double b);

public double divide(double a, double b);

public double evaluate(String query);

}

**Network Manager**

package com.mangnaik.yogesh.server;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.Socket;

public class NetworkManager {

final private DataInputStream dis;

final private DataOutputStream dos;

final private Socket socket;

public NetworkManager(DataInputStream dis, DataOutputStream dos, Socket socket){

this.dis = dis;

this.dos = dos;

this.socket = socket;

}

public String receive(){

String received;

try {

received = dis.readUTF();

System.out.println("Received String : " + received);

return received;

} catch (IOException e) {

System.out.println("Client Disconnected");

try {

socket.close();

return "";

} catch (IOException e1) {

e1.printStackTrace();

}

e.printStackTrace();

}

return "";

}

public void send(String answer){

try{

System.out.println("Sending String" + answer);

dos.writeUTF(answer);

}

catch(IOException e){

e.printStackTrace();

}

}

}

**Server.java**

package com.mangnaik.yogesh.server;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

/\*\*

\* Created by Yogesh on 2/8/2018.

\*/

public class Server {

ServerSocket socket;

Calculator calculator;

DataInputStream dis;

DataOutputStream dos;

NetworkManager networkManager;

public Server(){

calculator = new Calculator();

System.out.println(calculator.evaluate("5+6"));

start();

}

public void start(){

try {

socket = new ServerSocket(8192);

} catch (IOException e) {

e.printStackTrace();

System.out.println("Failed to create Server!!");

}

Socket s;

try{

s = socket.accept();

System.out.println("A new client has connected");

dis = new DataInputStream(s.getInputStream());

dos = new DataOutputStream(s.getOutputStream());

networkManager = new NetworkManager(dis, dos, s);

String query = "";

while(!query.equals("exit")){

query = networkManager.receive();

System.out.println("Inside query : " + query);

double answer = calculator.evaluate(query);

networkManager.send(answer + "");

}

} catch (IOException e) {

e.printStackTrace();

}

}

public static void main(String[] args){

new Server();

}

}