

Код: import java.io.\*;

import java.util.\*;

import java.util.regex.\*;

public class Calculator {

private static final String HISTORY\_FILE = "calculator\_history.txt";

private static List<String> history = new ArrayList<>();

public static void main(String[] args) {

loadHistory();

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("\nCalculator (enter 'exit' to quit)");

System.out.println("Available operations: + - \* / // % ^ |x|");

System.out.println("Enter equation or 'history' to view history:");

String input = scanner.nextLine().trim();

if (input.equalsIgnoreCase("exit")) {

break;

} else if (input.equalsIgnoreCase("history")) {

showHistory();

continue;

} else if (input.isEmpty()) {

continue;

}

try {

double result = evaluateExpression(input);

String equation = input + " = " + result;

System.out.println(equation);

addToHistory(equation);

} catch (Exception e) {

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

}

}

saveHistory();

scanner.close();

}

private static double evaluateExpression(String expression) {

// Обработка модуля |x|

expression = processAbsoluteValues(expression);

// Обработка степеней ^

expression = processExponents(expression);

// Обработка умножения, деления и остатка

expression = processMultiplicationDivisionModulo(expression);

// Обработка сложения и вычитания

return processAdditionSubtraction(expression);

}

private static String processAbsoluteValues(String expr) {

Pattern pattern = Pattern.compile("\\|([^|]+)\\|");

Matcher matcher = pattern.matcher(expr);

while (matcher.find()) {

double value = processAdditionSubtraction(matcher.group(1));

expr = expr.replace(matcher.group(), String.valueOf(Math.abs(value)));

}

return expr;

}

private static String processExponents(String expr) {

Pattern pattern = Pattern.compile("(-?\\d+\\.?\\d\*)\\^(-?\\d+\\.?\\d\*)");

Matcher matcher = pattern.matcher(expr);

while (matcher.find()) {

double base = Double.parseDouble(matcher.group(1));

double exponent = Double.parseDouble(matcher.group(2));

double result = Math.pow(base, exponent);

expr = expr.replace(matcher.group(), String.valueOf(result));

}

return expr;

}

private static String processMultiplicationDivisionModulo(String expr) {

Pattern pattern = Pattern.compile("(-?\\d+\\.?\\d\*)([\*/%//])(-?\\d+\\.?\\d\*)");

Matcher matcher = pattern.matcher(expr);

while (matcher.find()) {

double left = Double.parseDouble(matcher.group(1));

String op = matcher.group(2);

double right = Double.parseDouble(matcher.group(3));

double result = 0;

switch (op) {

case "\*":

result = left \* right;

break;

case "/":

if (right == 0) throw new ArithmeticException("Division by zero");

result = left / right;

break;

case "%":

result = left % right;

break;

case "//":

if (right == 0) throw new ArithmeticException("Division by zero");

result = (int)(left / right);

break;

}

expr = expr.replace(matcher.group(), String.valueOf(result));

}

return expr;

}

private static double processAdditionSubtraction(String expr) {

String[] parts = expr.split("(?=[+-])|(?<=[+-])");

double result = Double.parseDouble(parts[0]);

for (int i = 1; i < parts.length; i += 2) {

String op = parts[i];

double num = Double.parseDouble(parts[i+1]);

if (op.equals("+")) {

result += num;

} else if (op.equals("-")) {

result -= num;

}

}

return result;

}

private static void addToHistory(String equation) {

history.add(equation);

}

private static void showHistory() {

if (history.isEmpty()) {

System.out.println("History is empty");

return;

}

System.out.println("\nCalculation history:");

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

System.out.println((i+1) + ". " + history.get(i));

}

}

private static void loadHistory() {

File file = new File(HISTORY\_FILE);

if (!file.exists()) return;

try (BufferedReader reader = new BufferedReader(new FileReader(file))) {

String line;

while ((line = reader.readLine()) != null) {

history.add(line);

}

} catch (IOException e) {

System.out.println("Error reading history: " + e.getMessage());

}

}

private static void saveHistory() {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(HISTORY\_FILE))) {

for (String equation : history) {

writer.write(equation);

writer.newLine();

}

} catch (IOException e) {

System.out.println("Error saving history: " + e.getMessage());

}

}

}