

Porównanie Chat GPT do Gemini

Polecenie dla AI:

Napisz mi w Javie aplikację konsolowego kalkulatora naukowego

Chat GPT



ChatGPT

Oczywiście, oto prosty przykład kalkulatora naukowego napisanego w Javie:

java

Copy code

```
import java.util.Scanner;

public class ScientificCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double result;

        System.out.println("Witaj w kalkulatorze naukowym!");
        System.out.println("Dostępne operacje:");
        System.out.println("1. Dodawanie (+)");
        System.out.println("2. Odejmowanie (-)");
        System.out.println("3. Mnożenie (*)");
        System.out.println("4. Dzielenie (/)");
        System.out.println("5. Potęgowanie (^)");
        System.out.println("6. Pierwiastkowanie (√)");
        System.out.println("7. Sinus (sin)");
        System.out.println("8. Cosinus (cos)");
        System.out.println("9. Tangens (tan)");
        System.out.println("10. Wyjście");

        while (true) {
            System.out.print("Wybierz numer operacji: ");
            int choice = scanner.nextInt();
            if (choice == 10) {
                System.out.println("Do widzenia!");
                break;
            }
        }
    }
}
```

```
switch (choice) {
    case 1:
        result = add(scanner);
        break;
    case 2:
        result = subtract(scanner);
        break;
    case 3:
        result = multiply(scanner);
        break;
    case 4:
        result = divide(scanner);
        break;
    case 5:
        result = power(scanner);
        break;
    case 6:
        result = squareRoot(scanner);
        break;
    case 7:
        result = sine(scanner);
        break;
    case 8:
        result = cosine(scanner);
        break;
    case 9:
        result = tangent(scanner);
        break;
    default:
        System.out.println("Nieprawidłowy wybór.");
        continue;
}

System.out.println("Wynik: " + result);
}
scanner.close();
}
```

```
public static double add(Scanner scanner) {
    System.out.print("Podaj pierwszą liczbę: ");
    double num1 = scanner.nextDouble();
    System.out.print("Podaj drugą liczbę: ");
    double num2 = scanner.nextDouble();
    return num1 + num2;
}

public static double subtract(Scanner scanner) {
    System.out.print("Podaj pierwszą liczbę: ");
    double num1 = scanner.nextDouble();
    System.out.print("Podaj drugą liczbę: ");
    double num2 = scanner.nextDouble();
    return num1 - num2;
}

public static double multiply(Scanner scanner) {
    System.out.print("Podaj pierwszą liczbę: ");
    double num1 = scanner.nextDouble();
    System.out.print("Podaj drugą liczbę: ");
    double num2 = scanner.nextDouble();
    return num1 * num2;
}

public static double divide(Scanner scanner) {
    System.out.print("Podaj pierwszą liczbę: ");
    double num1 = scanner.nextDouble();
    System.out.print("Podaj drugą liczbę: ");
    double num2 = scanner.nextDouble();
    if (num2 == 0) {
        System.out.println("Nie można dzielić przez zero.");
        return Double.NaN; // NaN - Not a Number
    }
    return num1 / num2;
}
```

```

public static double power(Scanner scanner) {
    System.out.print("Podaj podstawę: ");
    double base = scanner.nextDouble();
    System.out.print("Podaj wykładnik: ");
    double exponent = scanner.nextDouble();
    return Math.pow(base, exponent);
}

public static double squareRoot(Scanner scanner) {
    System.out.print("Podaj liczbę: ");
    double number = scanner.nextDouble();
    return Math.sqrt(number);
}

public static double sine(Scanner scanner) {
    System.out.print("Podaj kąt w stopniach: ");
    double angle = Math.toRadians(scanner.nextDouble());
    return Math.sin(angle);
}

public static double cosine(Scanner scanner) {
    System.out.print("Podaj kąt w stopniach: ");
    double angle = Math.toRadians(scanner.nextDouble());
    return Math.cos(angle);
}

public static double tangent(Scanner scanner) {
    System.out.print("Podaj kąt w stopniach: ");
    double angle = Math.toRadians(scanner.nextDouble());
    return Math.tan(angle);
}
}

```

Ten kalkulator posiada funkcje dodawania, odejmowania, mnożenia, dzielenia, potęgowania, pierwiastkowania, sinusa, cosinusa i tangensa. Został on napisany w sposób interaktywny, więc użytkownik może wybrać operację i wprowadzić odpowiednie dane.



```
ScientificCalculator x
↑ Witaj w kalkulatorze naukowym!
↓ Dostępne operacje:
1. Dodawanie (+)
2. Odejmowanie (-)
3. Mnożenie (*)
4. Dzielenie (/)
5. Potęgowanie (^)
6. Pierwiastkowanie (√)
7. Sinus (sin)
8. Cosinus (cos)
9. Tangens (tan)
10. Wyjście
Wybierz numer operacji: 1
Podaj pierwszą liczbę: 5
Podaj drugą liczbę: 2
Wynik: 7.0
Wybierz numer operacji: 2
Podaj pierwszą liczbę: 7
Podaj drugą liczbę: 2
Wynik: 5.0
Wybierz numer operacji: 3
Podaj pierwszą liczbę: 5
Podaj drugą liczbę: 5
Wynik: 25.0
Wybierz numer operacji: 4
Podaj pierwszą liczbę: 10
Podaj drugą liczbę: 2
Wynik: 5.0
Wybierz numer operacji: 5
Podaj podstawę: 3
Podaj wykładnik: 3
Wynik: 27.0
Wybierz numer operacji: 10
Do widzenia!

Process finished with exit code 0
```

Chat GPT zaprogramował bardzo prostą i działającą aplikację konsolową kalkulatora naukowego. Nie zastosował żadnych zaawansowanych funkcji, otrzymał polecenie w języku polskim i wiadomości konsolowe zaprogramował właśnie w takim języku. Dodatkowo dołączył krótki opis aplikacji.

Gemini



Java

```
import java.util.Scanner;

public class ScientificCalculator {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.print("Wprowadź wyrażenie matematyczne: ");
            String expression = scanner.nextLine();

            try {
                double result = evaluateExpression(expression);
                System.out.println("Wynik: " + result);
            } catch (Exception e) {
                System.out.println("Błąd: " + e.getMessage());
            }
        }

        private static double evaluateExpression(String expression) throws Exception {
            // TODO: Implement logic to evaluate the mathematical expression
            // This is a complex task and requires a detailed understanding
            // For demonstration purposes, we can replace this with a placeholder implementation
            System.out.println("Not implemented yet. Placeholder implementation");
            return 0.0;
        }
    }
}
```

Używaj kodu [z rozważą](#).



This code provides a basic structure for a scientific calculator application in Java. It includes a main loop that prompts the user for an expression, evaluates the expression using a placeholder function, and prints the result.

The actual implementation of the `evaluateExpression` function is left as a TODO, as it involves a complex task of parsing and evaluating mathematical expressions. This would require a detailed understanding of parsing and evaluation algorithms. For a complete implementation, you would need to develop code that can handle various mathematical operations, functions, and operator precedence.

If you're interested in learning more about parsing and evaluation algorithms, there are many resources available online and in libraries. You can also find examples of existing expression evaluators in various programming languages.



Gemini nie zaprogramował całej aplikacji, stworzył tylko szkielet aplikacji i wypisał co należy zrobić aby stworzyć taką aplikację konsolową. Dodatkowo mimo polecenia w języku polskim jego wiadomości zwrotne pojawiły się w języku angielskim, ale wiadomości konsolowe w języku polskim. Natomiast w szkielecie widać, że skorzystał z obsługi wyjątków, której Chat GPT nie zastosował. Poprosiłem go aby zaimplementował logikę dla funkcji `evaluateExpression`.



Java

```
import java.util.Scanner;
import java.util.Stack;

public class ScientificCalculator {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.print("Wprowadź wyrażenie matematyczne: ");
            String expression = scanner.nextLine();

            try {
                double result = evaluateExpression(expression);
                System.out.println("Wynik: " + result);
            } catch (Exception e) {
                System.out.println("Błąd: " + e.getMessage());
            }
        }
    }

    private static double evaluateExpression(String expression) throws Exception {
        // Stack to store operands and operators
        Stack<Double> operands = new Stack<>();
        Stack<Character> operators = new Stack<>();

        // Tokenize the expression
        String[] tokens = expression.split(" ");

        for (String token : tokens) {
            if (isOperand(token)) {
                operands.push(Double.parseDouble(token));
            } else if (isOperator(token)) {
                char operator = token.charAt(0);

                while (!operators.isEmpty() && precedence(operator) <= precedence(operators.peek())) {
                    double rightOperand = operands.pop();
                    double leftOperand = operands.pop();
                    double result = applyOperator(operators.pop(), leftOperand, rightOperand);
                    operands.push(result);
                }

                operators.push(operator);
            }
        }

        if (!operators.isEmpty()) {
            throw new Exception("Invalid expression: missing closing parenthesis");
        }

        if (operands.isEmpty()) {
            throw new Exception("Invalid expression: missing operand");
        }

        return operands.pop();
    }

    private static boolean isOperand(String token) {
        return token.matches("-?\\d+(\\.\\d+)?");
    }

    private static boolean isOperator(String token) {
        return token.matches("[+*^-/() ]");
    }

    private static int precedence(char operator) {
        switch (operator) {
            case '(':
                return 1;
            case '+':
            case '-':
                return 2;
            case '*':
            case '/':
                return 3;
            case '^':
                return 4;
            default:
                return 0;
        }
    }

    private static double applyOperator(char operator, double left, double right) {
        switch (operator) {
            case '+':
                return left + right;
            case '-':
                return left - right;
            case '*':
                return left * right;
            case '/':
                return left / right;
            case '^':
                return Math.pow(left, right);
        }
        return 0;
    }
}
```

```

        operators.push(operator);
    } else {
        throw new Exception("Invalid token: " + token);
    }
}

// Evaluate remaining operators
while (!operators.isEmpty()) {
    double rightOperand = operands.pop();
    double leftOperand = operands.pop();
    double result = applyOperator(operators.pop(), leftOperand,
    operands.push(result);
}

// Return the final result
if (operands.size() != 1) {
    throw new Exception("Invalid expression");
}

return operands.pop();
}

private static boolean isOperand(String token) {
    try {
        Double.parseDouble(token);
        return true;
    } catch (NumberFormatException e) {
        return false;
    }
}

private static boolean isOperator(String token) {
    return token.equals("+") || token.equals("-") || token.equals("*") || token.equals("/");
}

private static int precedence(char operator) {
    switch (operator) {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        default:
            return 0;
    }
}
}

```

3

```
private static double applyOperator(char operator, double leftOperand, double rightOperand) {
    switch (operator) {
        case '+':
            return leftOperand + rightOperand;
        case '-':
            return leftOperand - rightOperand;
        case '*':
            return leftOperand * rightOperand;
        case '/':
            return leftOperand / rightOperand;
        default:
            throw new IllegalArgumentException("Invalid operator: " + operator);
    }
}
```

4

Używaj kodu [z rozważą](#).



The provided code implements the `evaluateExpression` function to evaluate mathematical expressions. It uses two stacks to store operands and operators, and it applies operators according to their precedence rules.

Explanation of the code:

1. Stack Initialization:

- Two stacks are created: `operands` to store numerical values and `operators` to store mathematical operators (+, -, *, /).

2. Tokenization:

- The input expression is split into tokens using the `split()` method, separating operands and operators based on spaces.

3. Expression Processing:

3. Expression Processing:

- Each token is processed:
 - **Operands:**
 - If a token is an operand (a numerical value), it is converted to a double using `Double.parseDouble()` and pushed onto the `operands` stack.
 - **Operators:**
 - If a token is an operator:
 - While the `operators` stack is not empty and the current operator's precedence is less than or equal to the precedence of the top operator in the stack:
 - Pop two operands from the `operands` stack.
 - Apply the top operator to the popped operands, storing the result back onto the `operands` stack.
 - Push the current operator onto the `operators` stack.
 - **Invalid Tokens:**
 - If a token is not an operand or operator, an `InvalidTokenException`

Źródła



1. github.com/rodrigopsasaki/math-engine
2. github.com/Gokulviper/MY-Ds-and-Algos
3. github.com/40067259/leetCode
4. github.com/PTrebor/calculatorMaven

```
ScientificCalculator x
↑ Wprowadź wyrażenie matematyczne: 5 + 2
↓ Wynik: 7.0
⌵ Wprowadź wyrażenie matematyczne: 5 - 2
Wynik: 3.0
⇄ Wprowadź wyrażenie matematyczne: 5 * 5
Wynik: 25.0
⇄ Wprowadź wyrażenie matematyczne: 15 / 3
Wynik: 5.0
Wprowadź wyrażenie matematyczne: 0
Wynik: 0.0
Wprowadź wyrażenie matematyczne: a
Błąd: Invalid token: a
Wprowadź wyrażenie matematyczne: |
```

Kod napisany przez Gemini wygląda już na bardziej zaawansowany niż ten napisany przez Chat GPT, dodatkowo w odpowiedzi zawarte zostało wyjaśnienie kodu, a także źródła z których Gemini korzystał. Klikając w numerki po lewej stronie kodu wyskakuje powiadomienie z jakiego źródła ten kod pochodzi. W kodzie nie zamieszczono jednak opcji zakończenia procesu z poziomu konsoli.

Podsumowanie:

Chat GPT nadaje się bardziej do pisania prostych i zrozumiałych aplikacji niż Gemini. Jeśli musimy napisać kod w języku programowania, którego nie znamy, a jednocześnie chcemy w miarę wiedzieć co ten kod robi to lepiej zastosować Chat GPT. Gemini jest lepsze do pisania bardziej zaawansowanych aplikacji. W odróżnieniu od Chat GPT, Gemini podaje źródła z których korzysta podczas tworzenia kodu aplikacji.