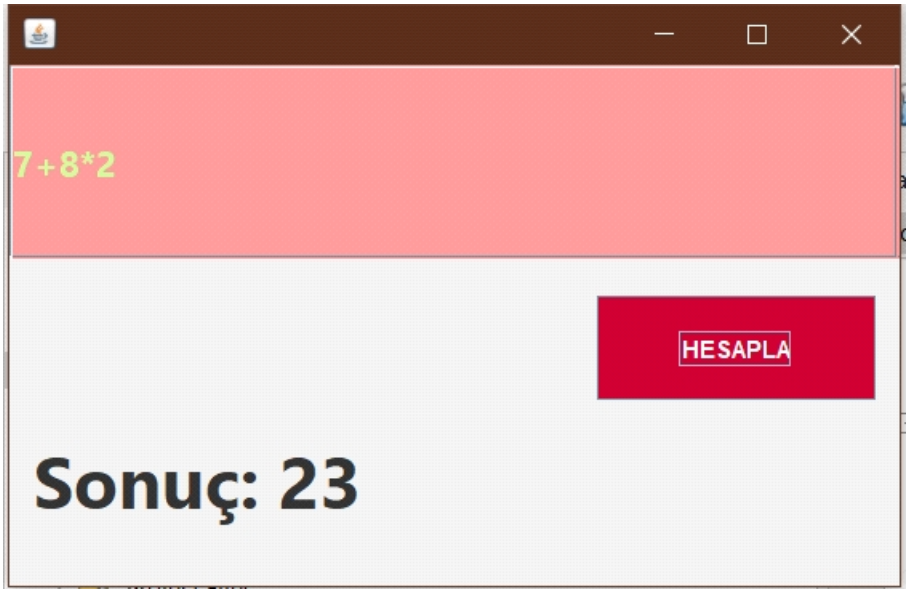
PROGRAMLAMA DİLLERİ ÖDEV-1



BNF TANIMI ;

<expression> ::= <term> | <expression> "+" <term>

<term> ::= <factor> | <term> "*" <factor>

<factor> ::= "(" <expression> ")" | <number>

<number> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9

AÇIKLAMALAR:

Expression, bir aritmetik ifadenin toplama işlemleri içeren en üst düzey yapısını tanımlar.

Term, çarpma işlemlerini içeren bir yapı tanımlar.

Factor, basit bir sayı veya parantezler içindeki bir ifade olabilir.

Number bir basit rakamdır. Yalnızca 0-9 arasında tek basamaklı sayıları içerir.

1.MADDE;

<expression> ::= <term> | <expression> "+" <term> ifadesi;

Bir expression, ya tek bir term olabilir (örneğin, 5 veya 3 * 4 gibi),

ya da bir expression'ın ardına "+" ekleyip başka bir term gelmesiyle oluşturulur (örneğin, 2 + 3 * 4).

2.MADDE;

<term> ::= <factor> | <term> "*" <factor> ifadesi;

Bir term, ya tek bir factor olabilir (örneğin, 7 veya 2 gibi),

ya da bir term'in ardına "*" ekleyip başka bir factor gelmesiyle oluşturulur (örneğin, 3 * 4).

3.MADDE;

<factor> ::= "(" <expression> ")" | <number> ifadesi;

Eğer factor bir number ise, bu tek bir basit sayı (örneğin, 5 veya 8) olacaktır.

Alternatif olarak, factor parantez içinde başka bir expression da olabilir.

Bu, aritmetik işlemlerde parantez kullanımının önceliği belirlemesi için önemlidir (örneğin, (2 + 3) * 4).

4.MADDE;

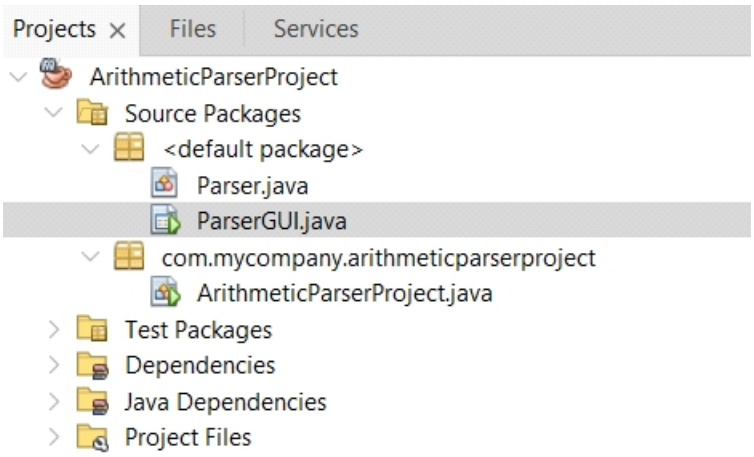
<number> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9

Bu kural, dilin daha karmaşık sayıları (örneğin, çok basamaklı tamsayılar) desteklemediğini,

yalnızca tek haneli rakamlarla işlem yapıldığını gösterir.

BU BNF yapısını anladıktan sonra işin javada gui arayüzüyle bu tanıma uygun parser yazdım

ArithmeticParserProject adında proje açtım içine jFrame Form ekledim



Design kısmına 1 jTextField 1 jButton ve 1 jLabel ekledim

Parser classımın kodları şu şekilde:

```
public class Parser {

    private String input;

    private int pos;


    public Parser(String input) {
```

```
    this.input = input;
    this.pos = 0;
}

// Toplama ve çarpma işlemlerini parse eden metot
public int parseExpression() {
    int value = parseTerm();
    while (pos < input.length() && input.charAt(pos) == '+') {
        pos++;
        value += parseTerm();
    }
    return value;
}

public int parseTerm() {
    int value = parseFactor();
    while (pos < input.length() && input.charAt(pos) == '*') {
        pos++;
        value *= parseFactor();
    }
    return value;
}

public int parseFactor() {
    if (input.charAt(pos) == '(') {
```

```
            pos++;
            int value = parseExpression();
            pos++;
            return value;
        } else {
            return parseNumber();
        }
    }

    public int parseNumber() {
        int start = pos;
        while (pos < input.length() && Character.isDigit(input.charAt(pos))) {
            pos++;
        }
        return Integer.parseInt(input.substring(start, pos));
    }
}
```

ParserGui classımın action performed kısmı ise şöyle:

```
    private void evaluateButtonActionPerformed(java.awt.event.ActionEvent evt) {

    String input = inputField.getText();
    Parser parser = new Parser(input);
    try {
        int result = parser.parseExpression();
```

```
        resultLabel.setText("Sonuç: " + result);
    } catch (Exception ex) {
        resultLabel.setText("Hatalı ifade!");
    }
```

SONUCU PDF'İN EN BAŞINDA GÖSTERDİM