Parser Simplificado - Questão 2

```
EXPR ::= EXPR + TERM | EXPR - TERM | TERM
TERM ::= TERM * UNARY | TERM / UNARY | UNARY
UNARY ::= + UNARY | - UNARY | POW
POW ::= FACTOR ^ UNARY | FACTOR
FACTOR ::= (EXPR) | ID | DIGIT ID ::= a | b | ... | z
DIGIT ::= 0 | 1 | ... | 9
```

A idéia primária é ajustar a gramática acima, logo após os ajustes ficará assim:

```
EXPR ::= TERM EXPR’
EXPR’::= +TERM EXPR’|-TERM EXPR’| ε
TERM::= UNARY TERM’
TERM’::= *UNARY TERM’| /UNARY TERM’| ε
UNARY::= + UNARY| - UNARY| POW
POW::= FACTOR POW’
POW’::= ^ UNARY POW’| ε
FACTOR::= (EXPR) | ID | DIGIT
ID ::= a | b | ... | z
DIGIT ::= 0 | 1 | ... | 9
```

Implementação na linguagem de programação

- No diretório **src\parser\IParser.java** encontra-se a interface do parser com nome **Iparser** .

No diretório *src\parser\Parser.java* encontra-se o desenvolvimento da interface

De início é necessário que haja seis métodos:

1. EOF
   - Serve para marcar fim de código.
2. Lookahead
   - Garante a visualização do próximo caracter para avançar na arvore sintática.
3. next
   - É responsável por avançar caracter por caracter do código fonte.
4. Match
   - Responsável por combinar os terminais e não terminais na arvore.
5. Error
   - Retorna a posição de erro na palavra.
6. Parser
   - Inicia-se com uma palavra de entrada que é armazenada em uma string no construtor .
   - cada caracter dessa palavra será passada de acordo com a gramática, caso não aceite ela não é exibida.

- Como regra a arvore inicia como expr no construtor, caso a string seja aceita a arvore é completada.

Como executar o programa

- No arquivo Main.java insira no vetor as palavra que queira testar.

```
package parser;

public class Main {
    public static void main(String[] args) {
       IParser parser = new Parser();

       String[] testStrings = {
            '''Insira aqui as palavras na inserção de vetor do tipo String, logo
após execute!

               exemplo -> a+b*c-d ou b^-2'''
       };

       for(String testeString : testStrings) {
            if(parser.parse(testeString)){
                System.out.println("String "+ testeString +" é valido");
            }
       }
    }
}
```