Design Document

1. Overview

本实验提供了一个完整的 Java 程序，它处理的对象是一种简单的中缀表达式：

- 表达式中的运算量只能是 0～9 的单个数字，仅支持加、减运算，且表达式中不允许任何空格、制表符等空白符号。
- 该程序的唯一功能是将终端用户输入的一个中缀表达式转换为等价的后缀表达式后输出，例如输入中缀表达式1+2，将输入其等价的后缀表达式12+。
- 若用户输入的表达式有误，则程序的输出是无意义的，但该程序不会提示用户输入有误。

通过修改以上程序，实现以下功能：

- 比较静态成员和非静态成员
- 比较消除尾递归前后程序的性能
- 扩展错误处理功能

运行环境

- **JDK:** 17+
- **JUnit:** 5.8.1 (用于单元测试)
- **开发工具:** 任意IDE (本项目在Eclipse上实现)

2. 实现步骤

2.1 静态成员与非静态成员

> 实验软装置中将类 Parser 中的数据成员（又称域，Field）int lookahead 定义为静态的（static），为什么要这样定义？ static成员属于类本身，因此该类的所有对象共享一个静态字段，如果不加static字段，表示每一个对象实例都有一个独立的lookahead成员。 使用静态变量可能是因为这样不需要考虑构造函数传参，可以直接共享变量。

> 尝试将 lookahead 定义为非静态的（即删除声明之前的修饰符 static），看看这是否会影响程序的正确性。

在代码中可以发现，只构造了一个Parser实例，因此是否使用static不会对实验结果造成影响。

```
public class Postfix {
    public static void main(String[] args) throws IOException {
        System.out.println("Input an infix expression and output its postfix notation:");
        new Parser().expr();
        System.out.println("\nEnd of program.");
    }
}
```

实验结果表明，确实如此。

- `static int lookahead` 测试结果

```
C:\Users\tcwl2\Desktop\SYSU_CourseWork\Compiler_Principle\Lab_1\LAB1_实验软装
_source>testctestcase.bat
Running Testcase 001: a correct input from DBv2.
================================================
The input is:
9-5+2
------------------------------------------------
Input an infix expression and output its postfix notation:
95-2+
End of program.
------------------------------------------------
The output should be:
95-2+
================================================
请按任意键继续

C:\Users\tcwl2\Desktop\SYSU_CourseWork\Compiler_Principle\Lab_1\LAB1_实验软装
_source>call testcase-002.bat
Running Testcase 002: a correct long input.
================================================
The input is:
1-2+3-4+5-6+7-8+9-0
------------------------------------------------
Input an infix expression and output its postfix notation:
12-3+4-5+6-7+8-9+0-
End of program.
------------------------------------------------
The output should be:
12-3+4-5+6-7+8-9+0-
================================================
请按任意键继续

C:\Users\tcwl2\Desktop\SYSU_CourseWork\Compiler_Principle\Lab_1\LAB1_实验软
Lab_1_source>call testcase-003.bat
Running Testcase 003: missing an operator.
================================================
The input is:
95+2
------------------------------------------------
Input an infix expression and output its postfix notation:

End of program.
------------------------------------------------
The output should be:
9 (error)
================================================
请按任意键继续

C:\Users\tcwl2\Desktop\SYSU_CourseWork\Compiler_Principle\Lab_1\LAB1_实验软
Lab_1_source>call testcase-004.bat
Running Testcase 004: missing an operand.
================================================
The input is:
9-5+-2
------------------------------------------------
Input an infix expression and output its postfix notation:
Exception in thread "main" java.lang.Error: syntax error
        at lab1.Parser.term(Postfix.java:90)
        at lab1.Parser.rest(Postfix.java:39)
        at lab1.Parser.rest(Postfix.java:46)
        at lab1.Parser.expr(Postfix.java:33)
        at lab1.Postfix.main(Postfix.java:134)
------------------------------------------------
The output should be:
95- (error)
================================================
请按任意键继续
```

- 改为 `int lookahead` 测试结果

```
C:\Users\tcwl2\Desktop\SYSU_CourseWork\Compiler_Principle\Lab_1\LAB1_实验软
Lab_1_source>build.bat
Compiling...
请按任意键继续

C:\Users\tcwl2\Desktop\SYSU_CourseWork\Compiler_Principle\Lab_1\LAB1_实验软
Lab_1_source>testcase.bat
Running Testcase 001: a correct input from DBv2.
================================================
The input is:
9-5+2
------------------------------------------------
Input an infix expression and output its postfix notation:
95-2+
End of program.
------------------------------------------------
The output should be:
95-2+
================================================
请按任意键继续

C:\Users\tcwl2\Desktop\SYSU_CourseWork\Compiler_Principle\Lab_1\LAB1_实验软
Lab_1_source>call testcase-002.bat
Running Testcase 002: a correct long input.
================================================
The input is:
1-2+3-4+5-6+7-8+9-0
------------------------------------------------
Input an infix expression and output its postfix notation:
12-3+4-5+6-7+8-9+0-
End of program.
------------------------------------------------
The output should be:
12-3+4-5+6-7+8-9+0-
================================================
请按任意键继续

C:\Users\tcwl2\Desktop\SYSU_CourseWork\Compiler_Principle\Lab_1\LAB1_实验软
Lab_1_source>call testcase-003.bat
Running Testcase 003: missing an operator.
================================================
The input is:
95+2
------------------------------------------------
Input an infix expression and output its postfix notation:
9
End of program.
------------------------------------------------
The output should be:
9 (error)
================================================
请按任意键继续

C:\Users\tcwl2\Desktop\SYSU_CourseWork\Compiler_Principle\Lab_1\LAB1_实验软
Lab_1_source>call testcase-004.bat
Running Testcase 004: missing an operand.
================================================
The input is:
9-5+-2
------------------------------------------------
Input an infix expression and output its postfix notation:
Exception in thread "main" java.lang.Error: syntax error
        at lab1.Parser.term(Postfix.java:90)
        at lab1.Parser.rest(Postfix.java:39)
        at lab1.Parser.rest(Postfix.java:46)
        at lab1.Parser.expr(Postfix.java:33)
        at lab1.Postfix.main(Postfix.java:134)
------------------------------------------------
The output should be:
95- (error)
================================================
请按任意键继续
95- (error)
================================================
请按任意键继续
```

可以发现，两种情况下程序的输出完全一致，不会对实验的正确性造成影响。

2.2 消除程序中的尾递归

- 尾递归思路
  - 当遇到 '+' 时：
    - 调用 `match('+')`跳过'+'。
    - 调用 `term()`处理后面的数字。
    - 输出 `'+'`符号。
  - 当遇到 '-' 时：
    - 调用 `match('-')`跳过'+'。
    - 调用 `term()`处理后面的数字。
    - 输出 `'-'`符号。
  - 当遇到其他字符时：
    - 终止。
- 消除尾递归思路 可以通过构建一个循环来判断当前的lookahead是否是'+'或者'-'：
  - 如果是，则执行相应操作。
  - 如果不是，就跳出循环。
- 代码实现

```
/**
 * Deal with the operands.
 *
 * When match '+' or '-':
 *    term() the following digit and output.
 * When match neither '+' nor '-':
 *    terminate.
 * @throws IOException
 */
void rest() throws IOException {
      while (lookahead == '+' || lookahead == '-') {
              int op = lookahead;
              match(op);
              term();
              System.out.write(op);
      }
}
```

- 实验结果

  用测试样例进行测试，发现实验结果与修改尾递归之前相同，说明修改后的`rest()`功能与改之前一致。

```
C:\Users\tcwl2\Desktop\SYSU_CourseWork\Compiler_Principle\Lab_1\LAB1_实验软装置\Lab_1_s    e
>build.bat
Compiling...
请按任意键继续

C:\Users\tcwl2\Desktop\SYSU_CourseWork\Compiler_Principle\Lab_1\LAB1_实验软装置\Lab_1_s    e
>testcase.bat
Running Testcase 001: a correct input from DBv2.
================================================
The input is:
9-5+2
------------------------------------------------
Input an infix expression and output its postfix notation:
95-2+
End of program.
------------------------------------------------
The output should be:
95-2+
================================================
请按任意键继续

C:\Users\tcwl2\Desktop\SYSU_CourseWork\Compiler_Principle\Lab_1\LAB1_实验软装置\Lab_1_s    e
>call testcase-002.bat
Running Testcase 002: a correct long input.
================================================
The input is:
1-2+3-4+5-6+7-8+9-0
------------------------------------------------
Input an infix expression and output its postfix notation:
12-3+4-5+6-7+8-9+0-
End of program.
------------------------------------------------
The output should be:
12-3+4-5+6-7+8-9+0-
================================================
请按任意键继续

C:\Users\tcwl2\Desktop\SYSU_CourseWork\Compiler_Principle\Lab_1\LAB1_实验软装置\Lab_1_s    e
>call testcase-003.bat
Running Testcase 003: missing an operator.
================================================
The input is:
95+2
------------------------------------------------
Input an infix expression and output its postfix notation:
9
End of program.
------------------------------------------------
The output should be:
9 (error)
================================================
请按任意键继续

C:\Users\tcwl2\Desktop\SYSU_CourseWork\Compiler_Principle\Lab_1\LAB1_实验软装置\Lab_1_s    e
>call testcase-004.bat
Running Testcase 004: missing an operand.
================================================
The input is:
9-5+-2
------------------------------------------------
Input an infix expression and output its postfix notation:
Exception in thread "main" java.lang.Error: syntax error
        at lab1.Parser.term(Postfix.java:108)
        at lab1.Parser.rest(Postfix.java:61)
        at lab1.Parser.expr(Postfix.java:45)
        at lab1.Postfix.main(Postfix.java:157)
------------------------------------------------
The output should be:
------------------------------------------------
The output should be:
```

2.3 为程序扩展错误处理功能

1. 实现思路

1. 划分错误类型

1. 通过自定义异常类 `ParseError`，通过枚举标识错误类型

```
class ParseError extends Exception {

    public enum ErrorType { LEXICAL, SYNTAX }
    private ErrorType type;
    private int position;

    public ParseError(String msg, ErrorType type, int pos) {
        super(msg);
        this.type = type;
        this.position = pos;
    }

    public ErrorType getErrorType() {
        return type;
    }
    public int getErrorPosition() {
        return position;
    }
}
```

2. **词法错误** 如果遇到不在合法输入范围内的字符（例如 &或空格），则在 `term()` 或 `rest()`内抛出异常时指定错误类型为 `ErrorType.LEXICAL`。

3. **语法错误** 当输入字符虽然是合法的（数字、+ 或 -），但其在语法上出现错误时，比如缺少操作数或运算符，程序会抛出语法错误。

    - 缺少运算符：在 `term()`中若在匹配一个数字后，紧跟着又是数字，则抛出**缺少运算符**的语法错误。
    - 缺少操作数：在 `term()`中若在期望操作数的位置遇到运算符，则抛出**缺少操作数**的错误。

4. **具体实现**

    1. `term()`

```
/**
  * Deal with the digits.
  * @throws IOException
  */
  void term() throws IOException, ParseError {
      if (Character.isDigit((char)lookahead)) {
          System.out.write((char)lookahead);
          match(lookahead);

          // Continuous digit is not allowed.
              if (lookahead != -1 && Character.isDigit((char)
lookahead)) {
                  throw new ParseError("Missing an operand",
```

```
                                        ParseError.ErrorType.SYNTAX,
position);
                }
            } else  {
                if (lookahead == '+' || lookahead == '-') {
                    throw new ParseError("Missing an operater",
                                        ParseError.ErrorType.SYNTAX,
position);
                }
                else {
                    throw new ParseError("Unexpected character: '" +
(char) lookahead + "'",
                                        ParseError.ErrorType.LEXICAL,
position);
                }
            }
    }
```

2. rest()

```
        /**
     * Deal with the operands.
    *
    * When match '+' or '-':
    *   term() the following digit and output.
    * When match neither '+' nor '-':
    *   terminate.
    * @throws IOException
    */
    void rest() throws IOException, ParseError {
        while (lookahead == '+' || lookahead == '-') {
            int op = lookahead;
            match(op);

        }
        if (lookahead == '\n'|| lookahead == '\r' || lookahead ==
-1) {
            return;
        }
        // If read an illegal character or space, report as an
error.
        //        System.out.println("Current char:" + lookahead);
        throw new ParseError("Unexpected character: '" +
(char)lookahead + "'",
                                ParseError.ErrorType.LEXICAL,
position);

    }
    ```
```

2. 错误定位
```

1. 通过一个 `position`变量记录当前输入位置，并且在输出报错时提供该位置信息。

3. 错误恢复

    1. `expr()`函数使用while循环读取的方式，可以在抛出异常之后继续扫描。
    2. 用try-catch语句进行报错检测，发现错误后，不终止程序，继续向下扫描。注意在 `expr()`函数中统一进行try-catch操作，`rest()`和 `term()`只抛出，不catch。这样可以使得每次抛出异常后，重新返回 `expr()`的循环开头，重新读取中缀表达式。
    3. 通过 `skip()`函数实现跳过异常字符的功能。

2. 测试结果

通过修改和添加testcase中的测试样例，验证上述功能的实现效果。

1. **修改后的测试样例**

    1. Testcase 001: a correct input from DBv2.

        1. input：9-5+2

        2. expected output: 95-2+

        3. 测试结果

```
Running Testcase 001: a correct input from DBv2.
================================================
The input is:
9-5+2
------------------------------------------------
Input an infix expression and output its postfix notation:
95-2+
End of program.
------------------------------------------------
The output should be:
95-2+
================================================
请按任意键继
```

    2. Testcase 002: a correct long input.

        1. input：1-2+3-4+5-6+7-8+9-0

        2. expected output: 12-3+4-5+6-7+8-9+0-

        3. 测试结果

```
Running Testcase 002: a correct long input.
================================================
The input is:
1-2+3-4+5-6+7-8+9-0
------------------------------------------------
Input an infix expression and output its postfix notation:
12-3+4-5+6-7+8-9+0-
End of program.
------------------------------------------------
The output should be:
12-3+4-5+6-7+8-9+0-
================================================
请按任意键继
```

3. Testcase 003: missing an operator & unexpected symbol.

    1. input：95+2

    2. expected output: 9 (error:SYNTAX) (error:LEXICAL) 2

    3. 测试结果

```
Running Testcase 003: missing an operator and unexpected symbol.
================================================
The input is:
95+2
------------------------------------------------
Input an infix expression and output its postfix notation:
9
Error Type: SYNTAX
Position: 2
Message: Missing an operand

Error Type: LEXICAL
Position: 3
Message: Unexpected symbol: '+'
2
End of program.
------------------------------------------------
The output should be:
9 (error:SYNTAX) (error:LEXICAL) 2
================================================
请按任意键继
```

    4. 解释

        - pos=1：（开始读取）读取9，预期数字，无错误，输出9
        - pos=2：读取5，预期运算符，SYNTAX错误，输出error（**回到 `expr()` 循环开头**）
        - pos=3：（开始读取）读取+，预期数字，LEXICAL错误，输出error（**回到 `expr()` 循环开头**）
        - pos=4：（开始读取）读取2，预期数字，无错误，输出2
        - pos=5：读取换行，结束。

4. Testcase 004: missing an operand & unexpected chatacter.

    1. input：9-5+-2

    2. expected output: 95- (error:SYNTAX) 2

    3. 测试结果

```
Running Testcase 004: missing an operand.
================================================
The input is:
9-5+-2
------------------------------------------------
Input an infix expression and output its postfix notation:
95-
Error Type: SYNTAX
Position: 5
Message: Missing an operater
2
End of program.
------------------------------------------------
The output should be:
95- (error:SYNTAX) 2
================================================
请按任意键继
```

4. 解释

    - pos=1：（开始读取）读取9，预期数字，无错误，输出9
    - pos=2：读取-，预期运算符，无错误，看下一位
    - pos=3：读取5，预期数字，无错误，输出5-
    - pos=4：读取+，预期运算符，无错误，看下一位
    - pos=5：读取-，预期数字，SYNTAX错误，输出error（**回到 `expr()` 循环开头**）
    - pos=6：（开始读取）读取2，预期数字，无错误，输出2
    - pos=7：读取换行，结束。

5. Testcase 005: unexpected characters.

    1. input：1 8+4*9+4-6

    2. expected output: 1 (error:LEXICAL) 84+ (error:LEXICAL) 94+6-

    3. 测试结果

```
Running Testcase 005: unexpected characters.
================================================
The input is:
1 8+4*9+4-6
------------------------------------------------
Input an infix expression and output its postfix notation:
1
Error Type: LEXICAL
Position: 2
84+
Error Type: LEXICAL
Position: 6
Message: Unexpected character: '*'
94+6-
End of program.
------------------------------------------------
The output should be:
1 (error:LEXICAL) 84+ (error:LEXICAL) 94+6-
================================================

请按任意键继
```

    4. 解释

        - pos=1：（开始读取）读取1，预期数字，无错误，输出1

- pos=2：读取 ，预期运算符，LEXICAL错误，输出error（**回到`expr()`循环开头**）
- pos=3：（开始读取）读取8，预期数字，无错误，输出8
- pos=4：读取+，预期运算符，无错误，看下一位
- pos=5：读取4，预期数字，无错误，输出4+
- pos=6：读取*，预期运算符，LEXICAL错误，输出error（**回到`expr()`循环开头**）
- pos=7：（开始读取）读取9，预期数字，无错误，输出9
- pos=8：读取+，预期运算符，无错误，看下一位
- pos=9：读取4，预期数字，无错误，输出4+
- pos=10：读取-，预期运算符，无错误，看下一位
- pos=11：读取6，预期数字，无错误，输出6-
- pos=12：读取换行，结束。

2.4 为程序增加文档化注释

为代码添加注释。 在`src`文件夹下运行：

```
javadoc lab1/Postfix.java -d doc
```