1 Division

The division operator is implemented as described in Chapter 1.5 of [1].

2 Remainder

The remainder operator is implemented analoguously to the division operator, with the only real difference being that in `Scanner.getNextToken()`, the "%" token does not need to be treated separately, whereas the division symbol could indicate the start of a comment and special care needs to be taken. For this reason, the code in this part was based on how multiplication is handled.

3 Unary Addition

The unary addition operator is implemented analoguously to the unary negation operator, which was already present in the code. Its main difference from the binary operators described above is that instead of working in the `JBinaryExpression.java` file, changes need to be made to `JUnaryExpression.java`. There is no dedicated JVM instruction for unary addition (contrary to unary negation, which uses `INEG`); for this reason, we use the `NOP` instruction, which does nothing.

4 Palindrome

To check a string for palindromicity, one possible algorithm traverses the string in both directions simultaneously, continuing as long as both traversed parts are the same. If these substrings differ at some point, the function outputs an empty string and terminates, otherwise it continues until the pointers meet, and outputs the original string. As discussed in [2], this algorithm is optimal, from a complexity point of view. One thing to be mindful of is potential case issues; to handle this, the string is converted to lower case before performing the comparison.

In order to check whether the middle of the string has been reached, the less-than operator ($<$) had to be implemented, which was done similarly to the greater-than operator which was already present in the code base. Tests were also written for this operator.

References

[1] Bill Campbell, Swami Iyer, and Bahar Akbal-Delibaş. *Introduction to Compiler Construction in a Java World*. Chapman and Hall/CRC, Boca Raton, Florida, 2012.

[2] Fernando Pelliccioni. *Palindromes and more*. Components Programming, 2016.