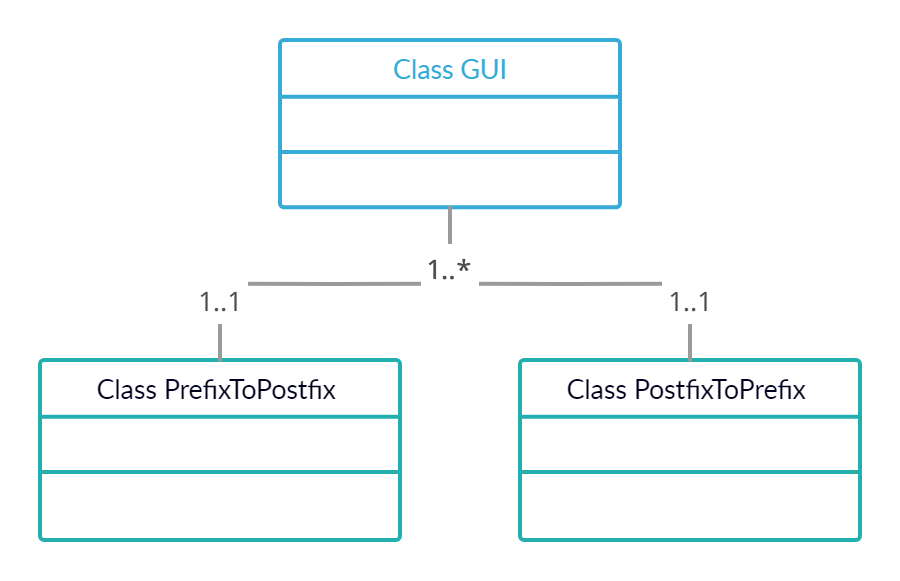
**UML Diagram**



**Test Cases**

1. 2 2 12 9 + 2 - + \* (Postfix to prefix with spaces)
2. 2 2 12 9+2-+\* (Postfix to prefix without spaces)
3. 2 2 129+2-+\* (Postfix to prefix gives empty stack)
4. \* 2 + 2 - + 12 9 2 (Prefix to postfix with spaces)
5. \*2+2-+12 9 2 (Prefix to postfix without spaces)
6. \*2+2-+12 92 (Prefix to postfix gives empty stack)

**Lessons learned**

It is also easier to translate the Postfix expression directly to Prefix without first translating it to Infix and then to Prefix in terms of computation and to better understand the expression.

The benefits both pre and postfix over conventional notation are essentially the same. These are the most significant ones:

1. A format that is ideal for direct execution is much simpler to translate. Any format can be transformed into a tree for further processing. Whether using a stack-base processor or a virtual computer, postfix can be directly transformed into code.
2. Infixed notation includes precedent and laws of associativity to disprove the notation or to add additional brackets that are normally not considered part of the notation. As long as any operator knows the number of arguments in advance, both prefix and postfix notations are absolutely unequivocal.
3. Easier for human developers to remember syntax rule.
4. Language implementations parse them easily and quickly.
5. They have ability to handle many operands.
6. They are simple to read, once you match a left parenthesis with its right one.