Documentation of project implementation for 1. task IPP 2018/2019
Name and surname: Dominik Vagala
Login: xvagal00

## Task

The goal was to create script, which will check if given IPPcode19 code is lexically and syntactically correct and print it's xml representation to STDOUT.

## Solution

Main script **parse.php** uses multiple classes from separate scripts to encapsulate logical parts and functions. Syntax analysis is not implemented by any strict scheme, because of relatively simple syntax in source language IPPcode19. All syntax rules for instructions are stored in array in this form: STRI2INT <var> <symb> <symb>, so if someone adds a new instruction to IPPcode19, only one line of code needs to be added. The main script provides following functionality:

1. Parse program arguments.
2. Check syntax on global level, call `InstructionsHelper::check_instruction_syntax()` to check individual instructions syntax.
3. Call `XmlGenerator::printXmlToStdin()` to generate xml code, if there's no lexical or syntactical error.

### Classes

**Token**
Represents token with type and data information.

**Lexer**
Provides lexical analysis. These steps are executed when `Lexer::nextToken()` is called:

1. Discard whitespaces and comments from stdin with `discardUselessStuffFromStdin()`, stop at the beginning of code or header.
2. If in previous step a new line character was discarded, return token object with EOL type. Otherwise continue.
3. Read symbols with `readString()`, stop at whitespace or comment.
4. With regexes determine what type of code was there, store the information in the token object and return to caller.

**Instruction**
Represent one instruction syntax, store information about instruction name and it's parameters. E.g., STRI2INT <var> <symb> <symb>.

**InstructionHelper**
Provides instruction syntax analysis. Functions:

- `fill_instructions_rules()`: fill array `instr_rules` with instructions syntax rules according to IPPcode19 specification.
- `get_instruction()`: return `instruction` object from array `instr_rules` by instruction name.
- `check_instruction_syntax()`: get current instruction name and gradually call `Lexer::nextToken()` to check if syntax is correct.

**XmlGenerator**
Wrapper for `Xmlwriter` extension and also prints actual xml code to STDOUT.