Implementační dokumentace k 2. úloze do IPP 2020/2021
Jméno a příjmení: *Yehor Pohrebniak*
Login: *xpohre00*

Interpret.py

1. **Description of interpret.py**
   1.1. Script interprets IPPcode21 with standard input and output from XML representation of code. Script implemented in file interpret.py. Main part of the program is implemented in function `main()`.
2. **Argument parsing**
   2.1. Argument parsing is implemented in function `argumentParse()` . Function checks if arguments are correct, and returns files with input and XML representation of IPPcode21.
3. **XML file parsing**
   3.1. Reading XML format is implemented in function `XMLParse()` which reads source file or standard input and generates ElementTree from library `xml.etree.ElementTree`. Function checks if XML format is correct. Function calls `checkHeader()` to control if XML header is correct and `checkChilds()` to generate a list of instructions sorted by their order and checks if indexes in arguments and order is not repeating. List with instructions is made in format `[order, instruction, (arg1, type, argumentBody), (arg2...]`.
4. **Instruction parsing**
   4.1. Parsing is implemented in function `parse()`. Function controls if instruction exists and if it is used with the correct arguments. Also it checks if the argument type matches the argument body.
5. **Interpretation of instructions**
   5.1. Interpretation is implemented in function `interpret()` which creates an object of a class `program`. Class `program` contains variables with an actual index in a programList, stack lists, frame lists.
   5.2. Initialization object of a class `program` will generate a list with the objects of a class `label` and check if labels do not repeat. Class `label` contains label's name and its index.
   5.3. Method `readInstruction()` in class `program` reads instruction with actual index and calls method with the same name as instruction. Every instructils method with the same name as instruction. Every instruction method will represent the instruction's description.
   5.4. Class `variable` contains variables with name, value and type. Object of this class is used for storing variables and conducting operations with them.

6. **Extensions**

    6.1. In the script were implemented extensions FLOAT, STACK. For implementation were created methods(stack instructions, float instructions) in the class `program` . Function for code parsing has been updated for types float and stack instructions.

Test.php

1. **Description of test.php**

    1.1. Script test.php exists for automatic testing interpret.py and parse.php. For testing should exist files with input code(ending .src), input representation in code(ending .in), output(ending .out), return value(ending. rc).

2. **Argument parsing**

    2.1. For argument parsing was created global variables `$directory`, `$parsefile`, `$intfile`, `$parseOnly`, `$intOnly`, `$recursive`, `$jexamxml`, `$jexamcfg`. Parsing is implemented in function `argParse()`, which sets global variables and checks if arguments are correct.

3. **Testing**

    3.1. Testing is implemented in function `test()`. Firstly it generates an array with all files in the directory with tests. With the loop it searches for the files with ending .src and starts testing scripts using function `exec()`. Compare exit codes with expected exit codes and compare outputs(diff or jexamXML file if parse-only).

4. **Generating HTML output**

    4.1. For generating output HTML page is using an object of class `HTMLGen`. It contains methods, which are generating variables with HTML code. Object creates and generates in function `test()`.

    4.2. HTML output represents in which mode (int-only, parse-only or both) scripts were tested, number of passed tests, failed tests, all tests and results of each test.