Documentation of project implementation for 2nd task IPP 2022/2023
Name and surname: Anastasiia Berezovska
Login: xberez04

Implementation features

Project is implemented using object-orienting programming in Python3.10.
Main class is `Execution`, its connect all other classes and execute the program in `execute()` method. For XML code parsing is used `xml.etree.ElementTree` library.

Project structure

Program is implemented in one script and library that consists of classes:
`interpret.py` – main program, argument parsing
`lib/`
- `Argument.py` – class of argument, attributes: type, value.
- `Instruction.py` – class of instruction and its methods.
- `Frame.py` – class of frame and its methods.
- `XMLParser.py` – class of xml parser and its methods.
- `Execution.py` – class of execution and its method.
- `error_code.py` – error codes, `exit_code()` function.

How program works

Main program is in the script `interpret.py`. After command line arguments are processed, `XMLParser` object is created and it calls `parse()` method that parses XML code, checks its semantic and stores instructions one by one in the instruction list. Instruction list consists of tuples: fisrt tuple item is instruction opcode, second is a list of tuples (each tuple represents an argument of instruction). XMLParser returns ordered (by instruction order) list of instructions in the `get_dict_ordered()` method. Then, `Execution` object is created and `execute()` method is called. In method `execute()` there is a loop which goes through instruction opcodes in instruction list and checks whether there is a match with the one of `IPPcode23` instructions, checking is implemented with `match()` function (is used from `re` module). When opcode is matched with some of the instructions, according methods are called, e.g. in `DEFVAR` instruction after setting argument attributes, `check_var_type()` is called to check if the type of current argument is correct, and if so, variable is defined. In case, there are some unexpected structures/operand types/etc, `exit_code()` function from `error_code.py` script is called, it prints message to `stderr` and exits program with the according code.

UML diagram shows project structure.