Implementation documentation for the IPP interpret project 2022/2023

Name and surname: Maryia Mazurava
Login: xmazur08

1 Description and program structure

Script `interpret.py` in Python 3.10 reads XML representation of the program and interprets an output according to the command line parameters.

Main program of the project is `interpret.py`, where input arguments are parsed in `parse_args()` function. The main arguments are:

- `--help` - prints a help message.
- `--source=`*`file`* - file with XML code.
- `--input=`*`file`* - file for possible test input.

One of the parameters `--source` or `--input` is always required. If one of them is missing, program reads from `stdin`.

After arguments are parsed method `parse()` from XMLParser class is called, which parses the source file using python module `xml.etree.ElementTree`. Here the object of class Program is created. Attributes of this class are instructions list and labels dictionary. Instructions list consists of objects of the Instruction class, while each object of this class consists of objects of the class Argument. Each object of different classes has own attributes.

At the end of file parsing the list of Instruction class objects is created, where all the instructions are sorted in ascending order. Method `execute()` of the Execution class iterates through the list of instructions and calls an appropriate instruction method if found.

Additional class Variable represents a variable in a program and contains it's name and value. Variables are stored in frames: GF - global frame (represented as dictionary with variable name as a key), LF - local frame, TF - temporary frame.

All the errors are handled with the function `error_exit()` in the `errors.py` file.

2 Implementation features

2.1 Labels and jumps

Labels names are collected and stored in a list during the parsing of the XML code.
All the "jump" instructions use the following algorithm:

1. Take the necessary order value of the label name key from the dictionary of labels.
2. Call the `execute()` method from class Execution with the order from the previous step as a parameter.
3. Now the `execute()` method iterates through the instructions list from the needed index. Default `current_order` variable is set to 0.

4. If the last instruction was executed, list of instructions is cleared and the **break** statement is used to terminate the loop so it doesn't continue to execute instructions from the place the "jump" instruction was executed.

2.2 Valid "order" attribute of the instructions

My implementation requires instruction attribute "order" in the XML code to start from 1 and without skipping any numbers. However, it is not required instructions to be sorted in ascending order in the XML file. For example: "2, 4, 3, 1, 5" is a valid order and "2, 5, 6, 11, 3" is not.

3 Class diagram