Implementační dokumentace k 2. úloze do IPP 2023/2024
Jméno a příjmení: Aryna Zhukava
Login: xzhuka01

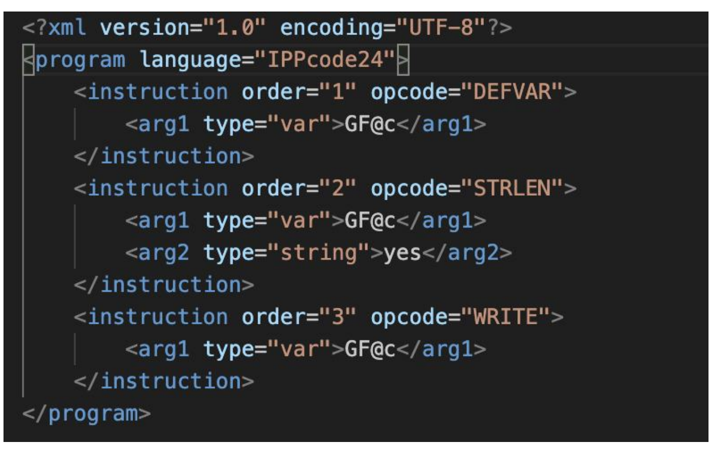
PHP XML INTERPRET

This program is an interpreter for files that contain programs written in the IPPcode24 language in XML representation.

Usage:
php interpret.php [[--source=[SOURCE_FILE]] [--input=[INPUT_FILE]]] [--help|-h]
--help - prints the help message
--source - specifies the path to the XML file that contains the program to be interpreted
--input - specifies the path to a file that contains the input data for the program

If source or input were not provided, the interpreter will wait for input from the standard input stream.

XML Format
The input XML file must conform to the following format:



```xml
<?xml version="1.0" encoding="UTF-8"?>
<program language="IPPcode24">
    <instruction order="1" opcode="DEFVAR">
        <arg1 type="var">GF@c</arg1>
    </instruction>
    <instruction order="2" opcode="STRLEN">
        <arg1 type="var">GF@c</arg1>
        <arg2 type="string">yes</arg2>
    </instruction>
    <instruction order="3" opcode="WRITE">
        <arg1 type="var">GF@c</arg1>
    </instruction>
</program>
```

The program element must have a `language` attribute with the value *IPPcode24*.
Each instruction element represents a single instruction in the program, and must have the following attributes:

'order' - an integer representing the order of the instruction in the program
'opcode' - a string representing the opcode of the instruction

Each instruction element may have up to 3 arg elements, each with the following attributes:

'type' - a string representing the type of the argument ("int", "string", "bool", "nil", "var", "type" or "nil")
'value' - the value of the argument, represented as a string

IMPLEMENTATION

Interpret.php

This class implements an interpreter for a custom instruction set. It reads an XML representation of instructions, executes them, and performs various operations based on the instructions.

public function execute(): int - is the main method of the whole program.
It is responsible for executing the instructions parsed from the input XML document.
It performs the following steps:

1. Initializes necessary variables and data structures.
2. Parses the XML document to extract instructions (*ParserXML.php*).
3. Validates the instructions and sorts them by order (*InstructionValidator.php)*.
4. Executes each instruction sequentially.
5. Handles various types of instructions such as arithmetic operations, control flow instructions, I/O operations, string operations, etc.
6. Returns an integer representing the exit code.

Literals.php

The class Symbol represents a symbol in the program. The 'value' property is set differently depending on the 'type'.
class Variable extends Symbol represents a variable, inheriting from the Symbol class. It contains additional properties for the variable's name($name) and scope($scope). The constructor extracts the scope and name from the variable's string representation.

Stack.php

Interface IStack declares getVariable() and declareVariable() methods.

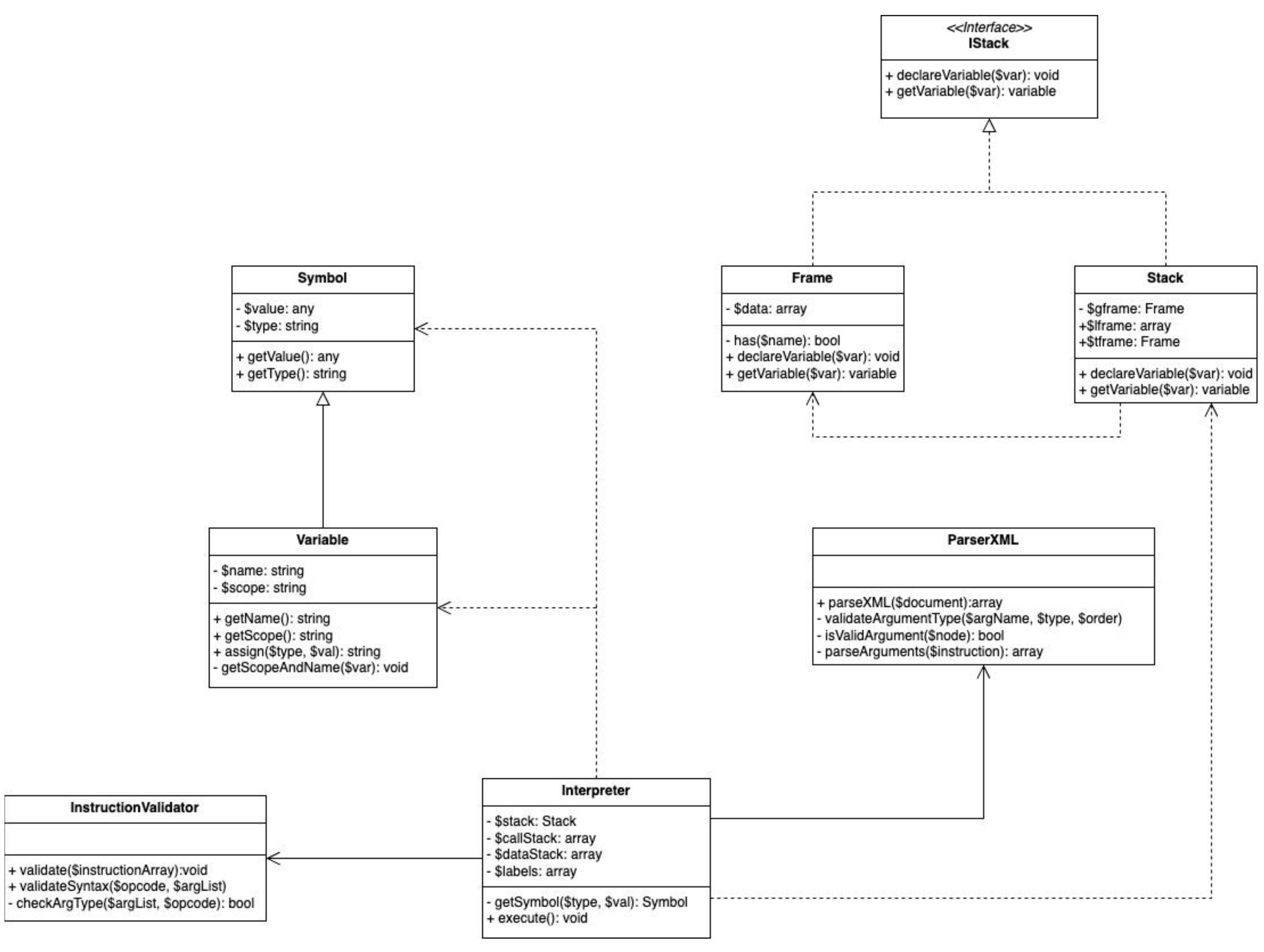
Class Frame implements IStack:
- Represents a frame within the variable scope hierarchy.
- Stores variables in an associative array.
- Provides methods to declare and retrieve variables within the frame.

Class Stack implements IStack:
- Manages the variable scopes including global, local, and temporary frames.
- Contains properties for global ($gframe), temporary ($tframe), and local ($lframe) frames.
- Provides methods to declare and retrieve variables within the appropriate frame based on scope (GF, LF, or TF).

Error Handling

Errors are reported to STDERR using the fwrite(STDERR, ...) function calls.
Error codes are validated using the **HelperFunctions::validateErrorCode()** method, that uses the IPP\core's return codes.

UML diagram


<<Interface>>
IStack
+ declareVariable($var): void
+ getVariable($var): variable
Symbol
- $value: any
- $type: string
+ getValue(): any
+ getType(): string
Frame
- $data: array
- has($name): bool
+ declareVariable($var): void
+ getVariable($var): variable
Stack
- $gframe: Frame
+$lframe: array
+$tframe: Frame
+ declareVariable($var): void
+ getVariable($var): variable
Variable
- $name: string
- $scope: string
+ getName(): string
+ getScope(): string
+ assign($type, $val): string
- getScopeAndName($var): void
ParserXML
+ parseXML($document):array
- validateArgumentType($argName, $type, $order)
- isValidArgument($node): bool
- parseArguments($instruction): array
InstructionValidator
+ validate($instructionArray):void
+ validateSyntax($opcode, $argList)
- checkArgType($argList, $opcode): bool
Interpreter
- $stack: Stack
- $callStack: array
- $dataStack: array
- $labels: array
- getSymbol($type, $val): Symbol
+ execute(): void