Implementační dokument k 2. úloze do IPP 2022/2023
Jméno a příjmení: Adam Nieslanik
Login: xniesl00

Po spuštění skriptu se zavolá třída `__main__`, ve které se ve smyčce volá metoda `run_instruction()` třídy `Run`.

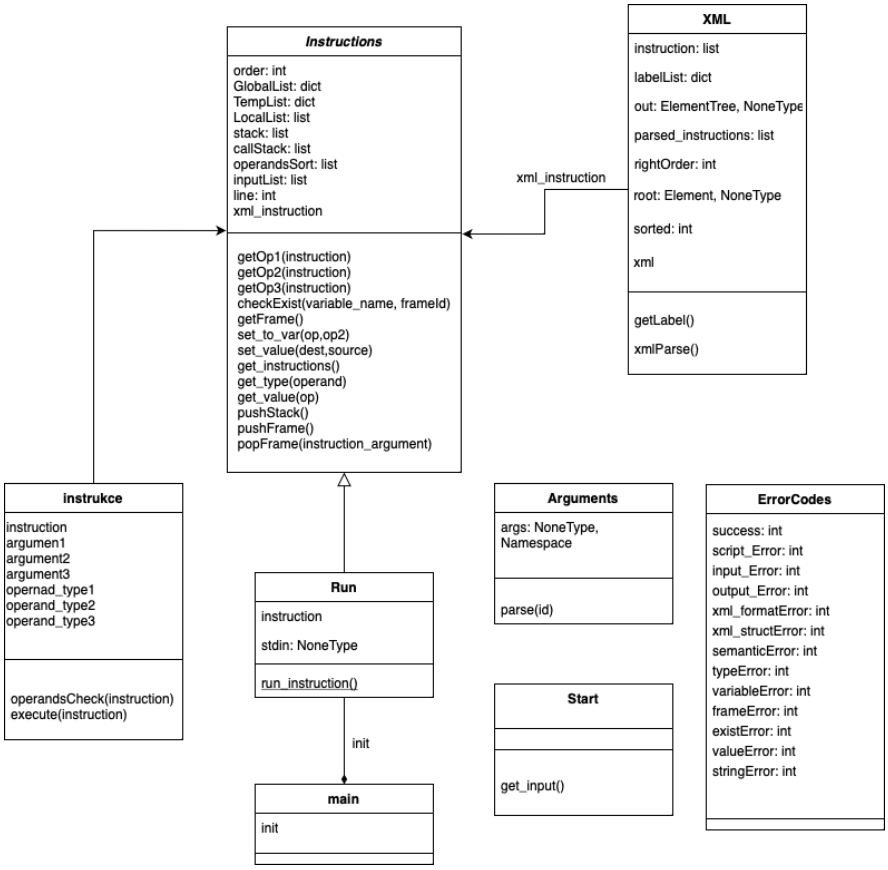
Třída `Instructions`, která je nadtřídou třídy `Run`, pří inicializaci použije metodu třídy Start, která vrátí načtený zdrojový soubor. Třída Start tento soubor získá pomocí třídy `Arguments`, ve které se zpracují argumenty příkazové řádky. Pokud třída `Arguments` nenalezne přepínač se zdrojovým souborem, jako zdrojový soubor vrátí soubor načtený ze standardního vstupu. Pokud však nebyl nalezen přepínač se vstupním souborem, vstupní soubor bude reprezentován standardním vstupem. Není ale možné nezadat alespoň jeden z těchto přepínačů. Skript podporuje spuštění s přepínačem `--help`, který vypíše na standardní výstup nápovědu k programu.
Pokud tedy třída `Start` vrátila načtený zdrojový soubor třídě `Instructions`, atribut `self.xml_instruction` této třídy bude inicializován jako třída `XML`, s načteným zdrojovým souborem. Následně se inicializuje atribut `self.instructionList`, který reprezentuje přijaty seznam instrukcí. Pro získání takového seznamu se použije metoda třídy `XML`, konkrétně `xml_parse()`, která načte `XML` reprezentaci kódu z předaného zdrojového souboru, a vrací seřazený seznam instrukcí, kde každá instrukce uchovává pořadí, svůj typ a jednotlivé argumenty. Na konec tohoto seznamu je přidán řetězec „END“, který značí, že další instrukce již seznam neobsahuje a slouží k správnému ukončení skriptu. Třída `Instructions`, reprezentuje tento seznam pomocí atributu `self.instructionList`. Pokud se podařilo načíst seznam instrukcí, může třída `Run`, pomocí metody `get_instruction()`, kterou dědí z nadtřídy `Instructions`, získat jednotlivou instrukci z tohoto seznamu. Pří zpracování jednotlivé instrukce se pomocí přepínače, který obsahuje hodnoty operačních kódů, volá metoda `execute()` z příslušící třídy. Každá instrukce je reprezentována svou vlastní třídou, všechny tyto třídy dědí metody a atributy ze třídy `Instructions`. Při interpretaci instrukce, se nejprve provede kontrola typů argumentů, a to pomocí metody `checkOperands()`, kterou vlastní jednotlivé třídy. Po kontrole typů se získá argumenty instrukce pomocí metody `getOp1()`, `getOp2()`, `getOp3()` ze třídy `Instructions`. Pokud je daný argument typu proměnná, zavolá se metoda `get_value()` z nadtřídy pro získání hodnoty této proměnné. Samotná interpretace probíhá v jednotlivých třídách, a pokud je výsledkem této operace přiřazení do proměnné, použije se metoda `self.set_to_var()` z nadtřídy pro zápis výsledku do proměnné v odpovídajícím rámci. Tato metoda pracuje z předanými parametry, kterými jsou jméno proměnné a místo jejího uložení (rámec) a kontroluje také, zda byla proměnná již definována. Tato kontrola se provádí pomocí metody `checkExist()`.

Skokové instrukce používají metodu `get_label()` z nadtřídy, která vrací slovník se jménem návěští a hodnotou, která popisuje pořadí instrukce v kódu. Tento slovník se naplní nalezenými návěštími již při zpracovávání zdrojového souboru ve třídě `XML`, aby bylo možné skočit na instrukci `LABEL`, která ještě nebyla zpracována. Hodnota získaná ze slovníku podle názvu návěští se přiřadí atributu `self.order` třídy `Instructions`, podle nějž se následné vybírá další zpracována instrukce.

Instrukce pracující s datovým zásobníkem používají zděděné metody jako `pushStack()` pro zápis hodnoty na zásobník, a `popStack()` pro získání hodnoty ze zásobníku. Datový zásobník je reprezentován atributem `self.stack` třídy `Instructions`. Pokud se jedná o instrukce, které pracují s rámci, používá se metody `getFrame()` pro získání hodnot rámce ze zásobníku rámců a `popFrame()` pro přesun rámce do dočasného rámce. Dočasný rámec představuje slovník, a obsahuje jméno „`init`“ s hodnotou `true/false`, která sděluje informaci o tom, zda byl dočasný rámec inicializován a je tedy možné k jeho hodnotám přistupovat či nikoli. Při zpracování instrukce `CALL` a `RETURN` je použit atribut `self.callStack` třídy `Instructions`, který slouží pro uchování hodnoty pořadí, na které se vrátí čítač pořadí `self.order` po vykonání instrukce `RETURN`.

Vstupně-výstupní instrukce jako je `READ/WRITE`, pracují se standardním vstupem či výstupem. Pokud byl zadán přepínač skriptu se vstupním souborem, instrukce `READ` použije tento soubor jako vstupní.

Většina z uvedených metod kontroluje správnost sémantiky a syntaxe a pří nalezení chyby ukončí program s příslušným návratovým kódem získaným ze třídy `ErrorCodes`.



Třída `instrukce` v UML diagramu reprezentuje všechny instrukce, avšak v implementaci má každá instrukce vlastní třídu. Z důvodu přehlednosti diagramu jsem zvolil sloučit tyto třídy do jedné.