Implementační dokumentace k 2.úloze do IPP 2023/2024

Jméno a příjmení: Jindřich Halva

Login: xhalva05

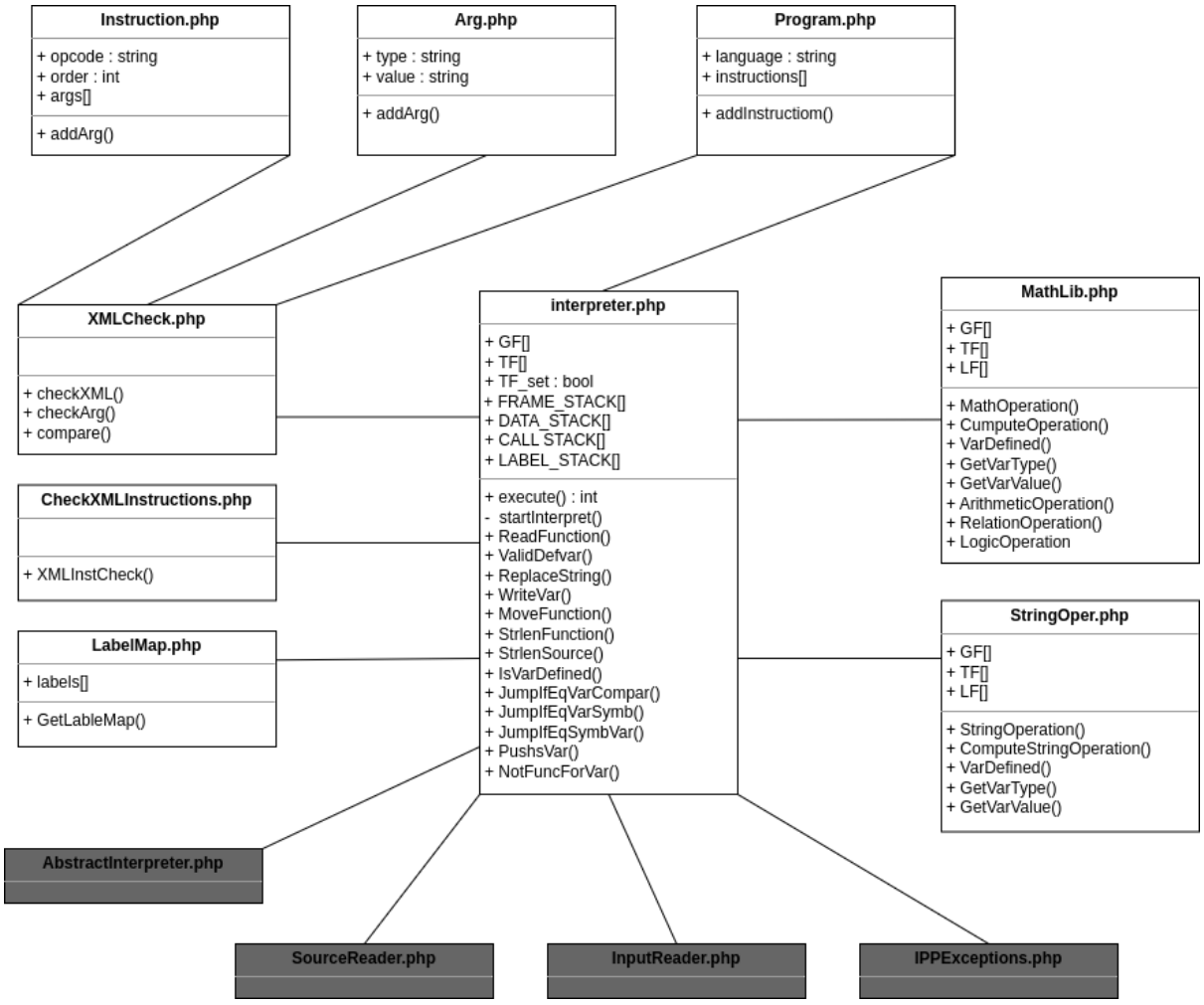
Úvod

Cílem druhé úlohy bylo vytvořit interpret pro program, reprezentován v XML. Požadavkem bylo využití jazyka PHP (verze 8.3) a užití objektově orientovaného programování. Program načítá zdrojový program v XML a vstup od uživatele, Výstup je směřován na standartní výstup.

K řešení byla poskytnuta základní kostra programu: ipp-core. Ošetřující vstupní parametry, načtení vstupního kódu atd.

Návrh

Následující UML diagram zobrazuje užité třídy. Každá třída uchovává dané atributy a metody. Šedou barvou jsou obarveny třídy, které komunikují s ostatními třídami, ale byly již implementovány rámcem ipp-core. Ke každé třídě se pojí ještě mnou vytvořené třídy spadající do kategorie “Exceptions”, (názvy souborů končí řetězcem “...Exceptions”) ty nejsou v diagramu zobrazeny pro lepší přehlednost. Názvy souborů jsou shodné s názvem třídy, kterou uchovává. Každý soubor má pouze jednu třídu.



Implementace

Soubor, který nese většinu funkčnosti interpretu se jmenuje `Interpreter.php`. Do tohoto souboru vstupuje program při zavolání funkce `execute()` ze souboru `Engine.php`.

Získaní zdrojového XML kódu programu zajišťuje rámec ippcore. Zdrojový program je uložen do proměnné a přeposlán funkci `checkXML()` z třídy `CheckXML`, která má za úkol zkontrolovat, zda je formát XML správný, či nalezne nějakou chybu ve struktuře programu. Pokud ano, program končí s návratovou hodnotou 32. Mezi chyby, které jsou odchyceny pomocí této třídy, patří například chybějící název instrukce, nebo opakující se čísla instrukcí. Interpret počítá s tím, že argumenty instrukcí jsou zapsány ve správném pořadí. Během funkce se plní třídy `Program`, `Instruction` a `Arg`. Třída `Program` je klíčová. Uchovává celou strukturu načteného programu a je tedy výstupem z této funkce. Před ukončením funkce jsou instrukce v tomto poli seřazeny podle jejich čísla `order`, seřazení probíhá ve vestavěné funkci `usort()`.

Pro kontrolu, zda jednotlivé instrukce obsahuji správný počet, typ a hodnotu argumentů existuje třída `CheckXMLInstructions`. Při kontrole je využíváno regulárních výrazů.

Před spuštěním samotné interpretace je potřeba ještě naplnit zásobník návěstí, resp. `LABEL_STACK`. O to se stará funkce z třídy `LabelMap`.

Poté začíná samotná interpretace. Program se k ní dostane přes funkci `StartInterpret()`. Program se prochází instrukci po instrukci. Pro danou instrukci se spustí daný úsek kódu nebo se zavolá funkce z nějaké jiné třídy. Většina instrukcí si ale vystačí bez volání cizích tříd a jsou naimplementovány přímo v hlavním souboru `Interpreter.php`. Typickou skupinou instrukcí, pro které je vytvořena speciální třída je skupina aritmetických, logických a relačních operací. Funkčnost těchto instrukcí nalezneme v souboru, pod stejnojmennou třídou, `MathLib.php`. Do této třídy vstupuje program přes funkci `MathOperation()`. Obdobně to platí pro skupinu instrukcí představující operace s řetězci (třída `StringOper`).

Dále byly vytvořeny třídy pro chybějící výjimky, které patří k daným návratovým kódům programu. Tyto třídy se nacházejí v souborech, jejichž název končí slovem "...Exceptions". Mezi tyto třídy patří následující:

- `XMLStructureException`
- `SemException`
- `OperandTypeException`
- `UnknownVarException`
- `FrameAccessException`
- `ValueException`
- `OperandValException`
- `StringOperationException`

V tomto interpretu chybí implementace instrukce `INT2CHAR`, která nebyla z časových důvodů zhotovena.

Program byl vyvíjen lokálně v editoru Visual Studio Code. K projektu byl poskytnut také kontejner s potřebným nastavením prostředí, ten byl řádně využit. Testování proběhlo na rozšířených testech, jejichž základy byly poskytnuty. K zajištění jisté kvality kódu bylo využito nástroje PHPStan. Projekt splňuje úroveň 6.