# Implementační dokumentace k projektu do IPP 2017/2018

Jméno a příjmení: Eliška Kadlecová
Login: xkadle34

## 1 Úvod

V tomto projektu jsem implementovala dva základní skripty parse.php a interpret.py, které zajišťují lexikální a syntaktickou analýzu jazyka včetně provedení samotných instrukcí jazyka IPPcode18. V následující části dokumentace Vám podrobněji popíši tyto zmiňované skripty jednotlivě.

## 2 Parse.php

Parse.php zajišťuje syntaktickou a lexikální analýzu jazyka IPPcode18. Skript ze standardního vstupu postupně ve smyčce čte instrukce v jazyce IPPcode18, které postupně analyzuje, případně vytváří XML soubor. Pokud vstupní kód obsahuje bílé znaky, mezery nebo komentáře, jsou tyto ignorovány, resp. odstraňovány.
Smyčka je implementována pomocí while cyklu, ve kterém v každé iteraci inkremenuji proměnnou `$num_line`, která uchovává informaci, který řádek se právě zpracovává. Tuto informaci potřebuji především na začátku, kdy se testuje, zda právě první řádek obsahuje řetězec ".IPPcode18"a pokud jej obsahuje, incializuje se XML soubor.
Jednotlivé instrukce jsem si rozřadila do skupin podle toho, kolik mají argumentů a jakých typů jsou. Podle toho s nimi pak dále pracuji, kontroluji, zda jsou zadány všechny argumenty dle očekávání a tvořím XML soubor na standardní výstup.
K rozhodování o tom, jaký je daný argument typ nebo zda hodnota odpovídá danému typu používám regulární výrazy definované na začátku skriptu.
XML soubor tvořím pomocí třídy DOMDocument.
Součástí mé implementace bylo i rozšíření STATP neboli statistický záznam o počtu komentářů a instrukcí. V každé iteraci smyčky se inkrementuje proměnná `$num_comments`, pokud se pomocí regulárního výrazu nalezne komentář. Čítač instrukcí (`$order_num`) se pak přičte pokaždé, co se ověří, že je zde další řádek s instrukcí. Úplně na konci (po uzavření stdin) se tyto informace uloží do zadaného souboru.

## 3 Interpret.py

Skript interpret.py provádí instrukce z daného XML souboru (výstupu předchozího skriptu).
Jako první zde provádím parsování zadaných argumentů a jejich kontrolu, následně otevírám a načítám data z XML souboru. K práci s XML využívám knihovnu `xml.etree.ElementTree`. Dále kontroluji hlavičku, která musí obsahovat informace o jazyku, a také kontroluji, zda všechny uvedené instrukce patří mezi instrukce jazyka IPPcode18 a zda mají správné označení `opcode` (tato čísla musí být od 1 postupně vzestupná, nesmí žádné chybět).
Po ověřování XML souboru se přechází k načtení a uložení všech návěští a k samotné interpretaci, která je implementována ve funkci `def interpret()`.
Intepret pracuje s několika globálními proměnnými (pro rámce, zásobníky, návěští), které byly inicializovány na začátku celého skriptu. Celou tuto funkci tvoří `for` cyklus, který prochází všechny podelementy kořenového elementu. Z nich se určí aktuální instrukce. U každé instrukce se vždy jako první provede kontrola na počet podelementů (neboli atributů). Dále se do pole `values[]` ukládají jednotlivé hodnoty argumentů (pokud je instrukce má). Toto pole se při každé iteraci cyklu vždy vymaže. Každý argument prochází kontrolami, které zjišťují, zda odpovídá uvedenému typu, v případě proměnné `var` se kontroluje i její případná existence v daném rámci.

# 4 Závěr

V tomto projektu jsem zvládla implementovat skript parse.php včetně rozšíření, dále pak interpret.py.
S jazyky PHP5.6 a Python3.6 jsem dosud neměla žádné zkušenosti, především pak z druhého z uvedených jsem měla poměrně obavy. Nakonec jsem se ale, po pročítání dokumentací, s oběma celkem zžila a jsem ráda, že jsem si oba mohla důkladněji "osahat".
Myslím, že jsem zadání splnila, ač jsem nestihla poslední, testovací, skript test.php.