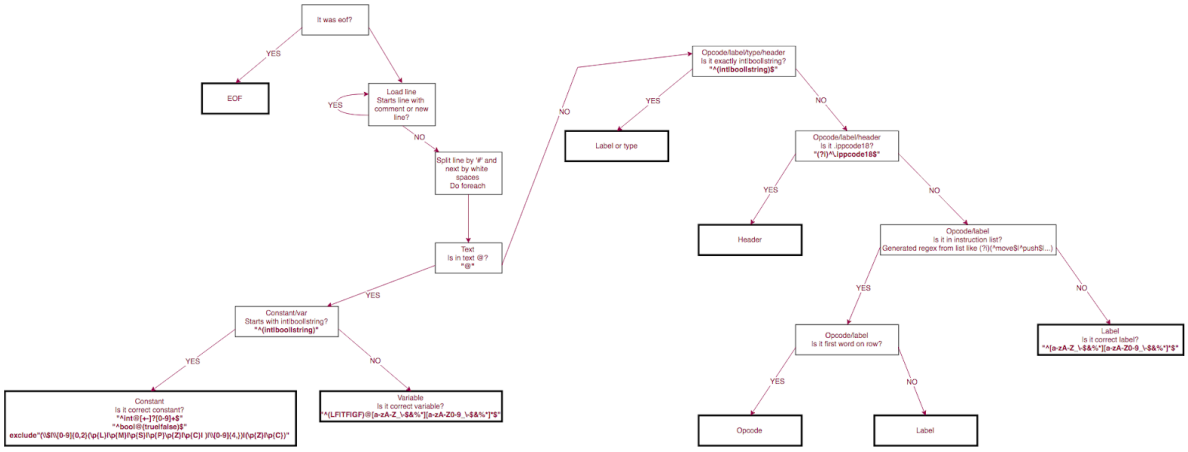
Implementační dokumentace k projektu do IPP 2017/2018

Jméno a příjmení: Jiří Juřica
Login: xjuric29

Parse.php

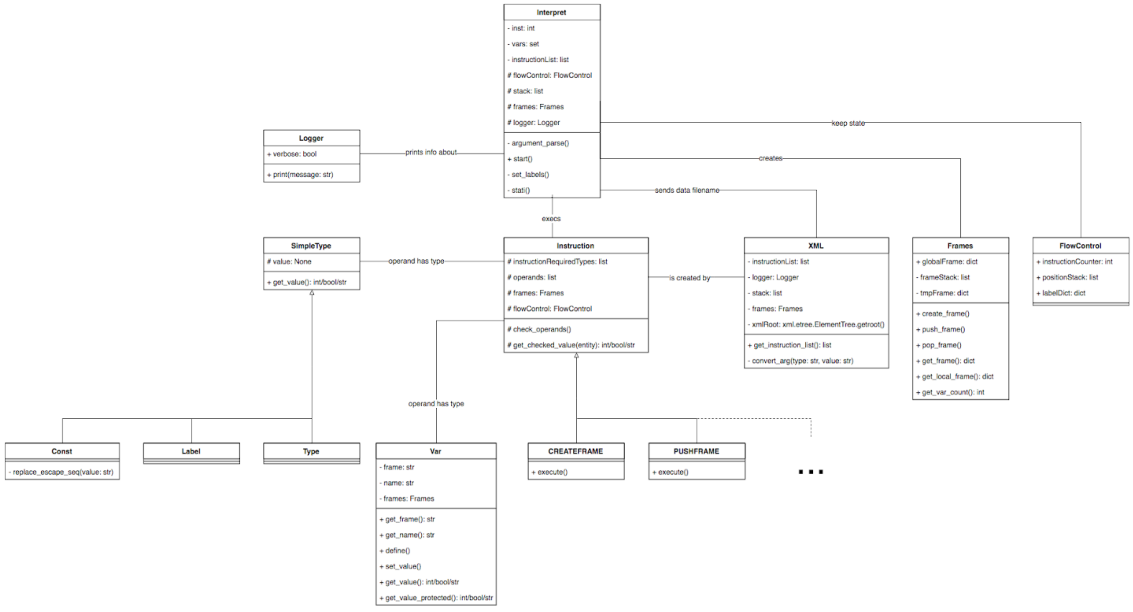
Skripty jsem se snažil strukturovat do logických částí pomocí oddělených souborů, proto se i tento skript skládá z hlavní části parse.php a následně souborů ve složce parse_lib scanner.php, syntax.php a others.php. V parse.php probíhá ošetření argumentů a případný zápis statistik z rozšíření do souboru a jinak už jen volá fci `syntaxer()` z syntax.php, který řídí celé zpracování.
Pro dobré vysvětlení funkčnosti syntaxeru je vhodné nejdřív přejít ke scanneru, který je v něm volán. Fce `scanner()` při každém jejím volání zpracuje jednu řádku užitečného kódu IPP18 (odřádkování a komentáře tedy ignoruje, dokud nenarazí na kód nebo eof) a rozdělí ji na opcode a argumenty (pokud se nejedná např. o hlavičku). V průběhu rozdělení kód zkontroluje po lexikální stránce. Zpracování funguje dle následujícího diagramu:



Převedená řádka na posloupnost tokenů je zpracována zmíněným syntaxerem, který ověří, zda se jedná o známou instrukci, odpovídá počet operandů a jejich typy. Nakonec pak tyto údaje převede jako součást výstupního xml.

Interpret.py

Skript importuje soubor modulů ze složky interpret_lib více, snažil jsem se na něm vyzkoušet si psaní objektově orientovaného kódu. Jednotlivé třídy jsou znázorněny na následujícím UML diagramu:



Instance třídy `Interpret` řídí celý běh programu. Nejdůležitější metodou je `start()`, která vytvoří instatnci třídy `XML`. Ta se postará o načtení, kontrolu a převedení veškerých částí kódů na předchystané objekty a `Interpretu` vrátí list po sobě jdoucích instancí instrukcí. V metodě `start()` je následně zavolána metoda `__set_labels()`, která před prováděním načte a zaznamená pozice pro instrukce `LABEL` a postará se přitom o jejich nahrazení za instrukce `NOP`. Následně již začne probíhat vykonávání kódu, které je ukončeno, načte-li se `None` z listu instrukcí, který slouží k zaznamenání konce.

Test.php

Po ošetření argumentů volá skript externí příkaz find, který řeší problém hledání (v daném adresáři nebo rekurzivně) jednotlivých testů. Find vyhledává soubory s příponou .src a při následném zpracování zjišťuje a případně doplňuje chybějící soubory.

Hlavní část skriptu sestavuje soubor dat pro každý test, které následně odesílá funkci addTest(data), která se stará o začlenění dat do vznikajícího html výstupu. Po skončení se vypíše html výstup, jehož součástí je tabulka s přehledem o tom, jak fungoval parse.php a interpret.php a zda se lišil výstup interpretu od referenčního, pokud oba skripty končili kódem 0.

Porovnání je opět prováděno externím programem diff a v případě, že je jeho výstup delší než tři řádky, v buňce se automaticky doplní javascriptový odkaz na zobrazení/skrytí celého obsahu.

Obecně

Pro každý skript platí, že u něj lze aktivovat debugovací režim, kdy na stderr vypisuje poměrně podrobně jeho aktuální činnost a lze pak jednoduše zjistit, v jaké části se v průběhu vykonávání vyskytl problém. Zapíná se pomocí připsání dlouhého argumentu --verbose u všech skriptů, krátké verze jsou různé v závislosti na tom, zda se -v nevyužívá již k něčemu jinému dle zadání.