Implementační dokumentace k 2. úloze do IPP 2018/2019

Jméno a příjmení: Peter Krutý
Login: xkruty00

1 Úvod

Cílem druhého úkolu bylo vytvořit 2 skripty, `interpret.py` v jazyce `python 3.6` a `test.php` v jazyce `PHP 7.3`. Skript `interpret.py` spočívá v načítání `XML` reprezentace IPPcode19 ze vstupního souboru / standardního vstupu a s využitím standardního vstupu / vstupního souboru ho interpretuje. Skript `test.php` slouží pro automatické testování postupné aplikace `parse.php` z první úlohy a `interpret.py` z druhé úlohy. Rovněž poskytuje možnost skripty testovat nezávisle. Tento fakt způsobuje, že je třeba přizpůsobit vstupní soubor s příponou `.src` podle toho, co se právě testuje. Kromě samotného otestování, skript generuje na standardní výstup přehledný souhrn o výsledcích testů ve formátu `HTML 5`.

2 Popis implementace skriptu `interpret.py`

Při návrhu skriptu `interpret.py` jsem využil znalosti získané na přednáškách předmětu IPP a rozhodl jsem se tak pro objektově-orientovaný návrh. Funkcionalitu jsem rozdělil do následujících 3 tříd: `ArgsParser`, `XMLParser`, `Interpret`. Pro každou z těchto tříd jsem vyčlenil samostatný soubor. V dalších odstavcích bude následovat jejich podrobnější popis.

2.1 `args_parser.py`

Soubor `args_parser.py` obsahuje třídu zapouzdřující metodu a atributy, které slouží pro zpracování vstupních argumentů skriptu. Konkrétně se jedná o metodu `parseArguments()` a atributy nesoucí informaci o dlouhých a krátkých argumentech. Zpracování je realizováno nástroji z modulu `getopt`.

2.2 `xml_parser.py`

V souboru s názvem `xml_parser.py` se nachází třída obsahující metody pro kontrolu syntaktické a lexikální správnosti. Princip spočívá v načítání XML kódu do proměnné `xml_root` pomocí funkce `xml.etree.ElementTree`. Při kontrole bylo dále třeba řešit řadu problémů jako například správné pořadí instrukcí. V `XML` je pořadí elementů nedefinované a tak je tento problém řešen explicitním přeuspořádáním.

2.3 `interpret.py`

Základní komponentou skriptu je modul `interpret.py`, který má na starosti samotnou interpretaci. Obsahuje hlavní kód skriptu a třídu `Interpret`, která pracuje s několika atributy (pro rámce, zásobníky, návěstní). Princip interpretace je založen na cyklickém procházení jednotlivých instrukcí, přičemž je každá kontrolována a interpretována podle potřeby.

2.4 `errors.py`

Modul `errors.py` zapouzdřuje definici chybových kódů, zpráv a funkci realizující odpovídající výpis na standardní chybový výstup.

2.4 Rozšíření `STATI`

V zadání jsem dostal možnost implementovat rozšíření skriptu. Rozšíření zahrnuje analýzu struktury zdrojového kódu v podobě počtu obsažených instrukcí a maximálního počtu inicializovaných proměnných. Více informací k dispozici v nápovědě samotného skriptu.

3 Popis implementace skriptu `test.php`

Implementace skriptu je rozdělena do 2 modulů. Moduly lze najít pod jmény `test.php`, `errors.php`. V dalších částech bude následovat jejich podrobnější popis.

3.1 `test.php`

V modulu test.php se nachází implementace automatických testů, kde každý test je reprezentován sadou 4 souborů (s příponami `.src`, `.in`, `.out`, `.rc`). Jestliže jeden ze souborů neexistuje, tak je automaticky vytvořen. Skript poskytuje 3 možnosti testování: testování `parse.php`, testování `interpret.py`, testování obou skriptů provázaně.

3.2 `errors.php`

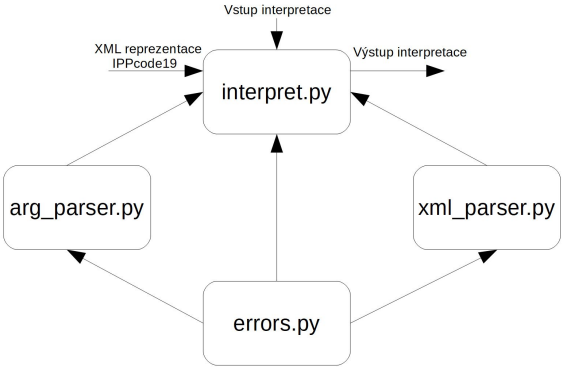
Modul `errors.php` zapouzdřuje definici chybových kódů, zpráv a funkci realizující odpovídající výpis na standardní chybový výstup.

4 Závěr

Druhý úkol byl programátorských velmi zajímavý protože jsem měl možnost si poprvé vyzkoušet práci v programovacím jazyce `Python` s množstvím zajímavých konstrukcí a modulů. Rovněž považuji za přínos realizaci skriptu `test.php`, nakolik se jednalo o moji první implementaci automatizovaných testů.

A Závislosti skriptů

Interpret



Automatizované testy