**Implementační dokumentace k projektu do IPP 2017/2018**

Jméno a příjmení: Martin Omacht

Login: xomach00

# PHP skripty

Skripty napsané v jazyce PHP (parse.php a test.php) využívají souboru autoload.php, který zajišťuje automatické načítání tříd ze souborů se stejným jménem jako daná třída ve složce classes/php/. Kromě výjimek, které jsou všechny v souboru Exceptions.php. Oba skripty pak mají každý svoji hlavní třídu (ParserApp a TesterApp), která dědí z abstraktní třídy App. Tato třída využívá návrhového vzoru Singleton, aby se kdekoliv z programu dalo přistupovat k objektu této třídy. Stará se o zpracování základní konfigurace programu z příkazové řádky a následné spuštění programu. Dále se však skripty již liší.

## Parse.php

Skript za pomocí tříd CodeAnalyzer a XMLOutput transformuje zdrojový soubor na výstupní soubor ve formátu XML. CodeAnalyzer funguje na bázi stavového automatu a využívá třídu IPPcode18 pro kontrolu a zpracování zdrojového kódu. Třida IPPcode18 obsahuje definice jazyka IPPcode18 a pomocné metody pro kontrolu jeho částí (operační kódy instrukcí, argumenty instrukcí, …). Skript očekává hned na prvním řádku hlavičku .IPPcode18, jinak je zdrojový soubor nesprávný. Ve výchozím nastavení aplikace pracuje se standardním vstupem a výstupem, to je však možné ovlivnit specifikováním vstupního souboru pomocí argumentu --src=soubor nebo výstupního pomocí --out=soubor.

Rozšíření STATP je řešeno pomocí návrhového vzoru Observer, kde třída StatisticsCollector implementuje rozhraní EventListener a naslouchá událostem, které nastanou ve třídě CodeAnalyzer (např. když je odstraněn komentář nebo je zpracovávána instrukce). Na základě těchto událostí pak sbírá statistiky. Pokud není zadán parametr --loc ani --comments, tak se vytvoří prázdný soubor.

Moduly skriptu jsou testovány pomocí knihovny PHPUnit. Tato knihovna poskytuje i jednoduché porovnávání řetězců s XML, proto byla použita i pro celkové testy tohoto skriptu.

## Test.php

Script testuje funkčnost obou skriptů parse.php a interpret.py najednou. Funguje jak na systému Windows, kde na porovnání výstupu používá příkaz FC, tak na systému Linux, kde je používán příkaz diff. Pro spouštění testovaných skriptů se ve výchozím nastavení používají příkazy php5.6 a python3.6, to lze však změnit parametry --php-int=soubor a --py-int=soubor. Dále se dá parametrem --text změnit výstup skriptu z HTML na obyčejný textový výstup a parametrem --temp-dir=adresář lze změnit umístění dočasných souborů při běhu aplikace, výchozí složka je aktuální složka.

Před samotným testováním se zkontroluje, zda existují všechny testované skripty. Poté se projde zadaný adresář s testy a vyhledají se všechny soubory s příponou .src. Nad každým takovýmto souborem se vytvoří instance třídy TestCase. Ta nejdříve nalezne zbylé soubory testu (s příponami .rc, .in a .out) nebo pokud neexistují, tak je vygeneruje. Následně zavolá testované skripty nad testovacími daty a porovná jejich výstupy s referenčními výstupy. Pokud tyto výstupy nesouhlasí, zaznamenají se detaily chyby do instance třídy TestResult, která se pak předá instanci třídy implementující rozhraní TestOutput (podle zadaného parametru buď TextTestOutput nebo HTMLTestOutput). Tato třída pak vypíše výsledky všech testů v požadovaném formátu.

# Interpret

V interpretu nejdříve převede třída IPPParser vstupní XML soubor na objektovou reprezentaci (pomocí ElementTree) a zkontroluje jeho správnost. Poté třída Program převede objektovou reprezentaci XML na seznam instrukcí (objekty třídy Instruction) a uloží si adresy všech návěstí. Následně zbývá samotná interpretace. Postupně se provádějí jednotlivé instrukce tím, že se volá jejich metoda run(). Tato metoda nejdříve zkontroluje, jestli jsou argumenty instrukce správného typu a jestli jsou inicializované. Pak zavolá funkci namapovanou k operačnímu kódu prováděné instrukce. Tyto funkce jsou mapovány pomocí dekorátoru. Zde je krátký příklad mapování funkce k operačnímu kódu STRLEN (funkce se mapuje na základě jména):

@Instruction.run\_func  
def \_strlen(context, dest: Arg, string: Arg):  
 dest.set\_value(context, len(string.get\_value(context)))

Takto definovaná funkce je pak uložená ve slovníku třídní proměnné třídy Instruction.

Interpret podporuje indexování řetězců pomocí záporných indexů stejně jako v jazyce Python. Při pokusu o redefinici proměnné se vytvoří nová neinicializovaná proměnná, která přepíše tu starou.