# Implementační dokumentace k 2. úloze do IPP 2020/2021

Jméno a příjmení: Tereza Burianová
Login: xburia28

## 1 Skript interpret.py

### 1.1 Základní funkcionalita

Účelem tohoto skriptu je kontrola, zpracování a interpretace kódu *IPPcode21* přeloženého do XML podoby skriptem *parse.php*. První část procesu, tedy zpracování a kontrolu XML souboru, zajišťuje třída *Preparation*. Metoda *fill_dictionary* naplní slovník informacemi o povolených instrukcích a jejich argumentech. Pomocí *argument_parse* se zpracují parametry příkazové řádky, tedy zdroj XML souboru *source* a případný vstupní soubor pro instrukci READ *input*. Následně se metoda *xml_parse* postará o parsování XML souboru a seřazení jeho značek podle atributu *order*, případně podle pořadí argumentů. Správnost vstupního souboru je kontrolována metodou *xml_validity*. Pro celé zpracování XML je využit modul *xml.etree.ElementTree*.

Hlavní částí skriptu je třída *Interpret*. První průchod kódem zajistí uložení návěští pomocí třídy *Labels*, druhý průchod už je pak samotná interpretace. Kromě pomocných metod obsahuje také odpovídající metody zpracovávající funkcionalitu každé z instrukcí. Značky <*instruction*> jsou postupně procházeny cyklem, který tyto metody dynamicky volá podle operačního kódu aktuální instrukce. Globální, lokální i dočasné rámce jsou všechny reprezentovány instancemi třídy *Frame*, která obsahuje *slovník* pro proměnné a pomocné metody *define_variable*, *edit_variable* a *get_var_value*. Lokální rámce jsou zpracovány jako zásobník těchto instancí.

### 1.2 Rozšíření STACK

Rozšíření je implementováno pomocí přídavných metod třídy *Interpret* reprezentujících zásobníkové verze některých instrukcí. Zpracování je téměř totožné, metody ale pracují s datovým zásobníkem *data_stack*, ze kterého si odeberou potřebný počet dat a přidají případný výsledek na jeho vrchol. Zásobníky jsou implementovány datovou strukturou *list*.

# 2 Skript test.php

## 2.1 Funkcionalita

Tento skript slouží k automatickému otestování skriptů *parse.php* a *interpret.py*. Cílem je vytvoření přehledné webové stránky s výsledky testů.

Třída *test_settings* zpracuje případné parametry skriptu, zadané uživatelem při jeho spouštění. Tyto jsou pak přepisovány metodou *fill_variables*. Pro případ chybějících nastavení již třída obsahuje výchozí hodnoty. Kontrola správnosti zadaných parametrů je provedena metodou *opts_validity*.

Samotný běh skriptů a porovnání hodnot zajišťuje třída *test_run*. Nejprve je pomocí *get_tests* vytvořeno pole obsahující veškeré vstupní soubory testů. Každý z těchto testů je pak zpracován odpovídajícími skripty (spuštěnými pomocí příkazu *exec*) v metodách *parse_only*, *int_only* a *both*. Výstup skriptu je (v případě odpovídajícího návratového kódu s hodnotou 0) porovnán pomocí nástrojů *diff* (výstup *interpret.py*) a *A7Soft JExamXML* (výstup *parse.php*) v metodách *check_out* a *check_out_xml*. Podrobněji je průběh testování, tedy skripty, jejich vstupy, výstupy a chování při různých nastaveních, vyobrazen diagramem na straně 3.

Generování HTML stránky probíhá ve třídě *generate_HTML*. Nalevo stránka obsahuje oddíl s informacemi o nastavení aktuálního běhu testování. Nová odrážka může být přidána voláním *create_setting*. Napravo se pak nachází další oddíl s celkovými výsledky tohoto běhu, včetně procentuálního hodnocení, vytvořený metodou *add_stats* na samotném konci testování. Nový řádek pro úspěšný či neúspěšný výsledek přidají po každém jednotlivém testu metody *add_success* a *add_fail*. Každý takový záznam obsahuje pořadí testu, absolutní cestu jeho zdroje a informaci o jeho úspěšnosti/selhání. Všechny oddíly jsou pak složeny dohromady a vypsány na standardní výstup metodou *print_html*.

## 2.2 Diagram popisující implementaci