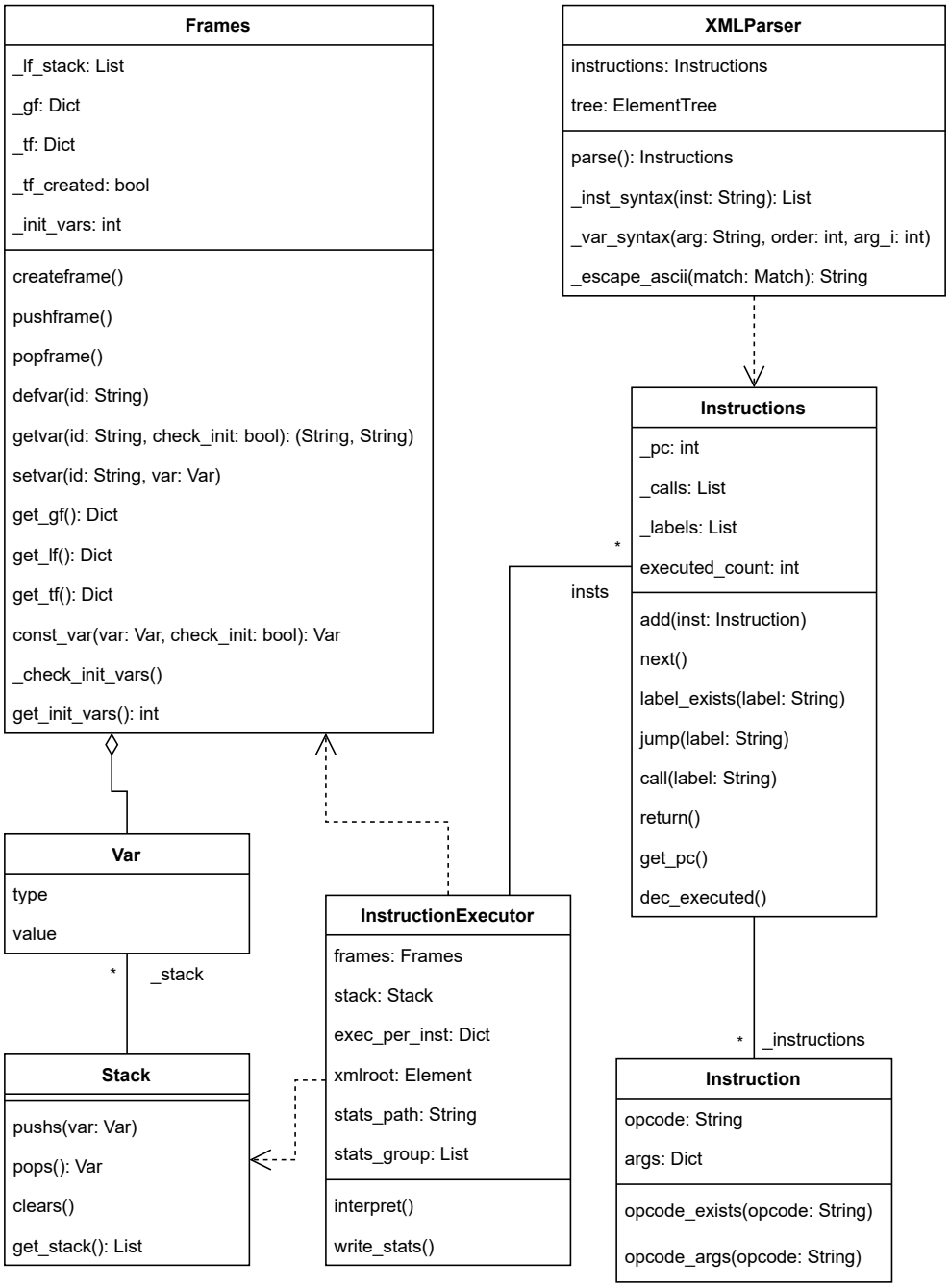
Implementační dokumentace k 2. úloze do IPP 2020/2021
Jméno a příjmení: Petr Kabelka
Login: xkabel09

1 Interpret

Skript *interpret.py* je rozdělen do několika tříd které mají na starosti jednotlivé části interpretu. Vztahy mezi jednotlivými třídami jsou znázorněny na následujícím diagramu.



Obrázek 1: Diagram tříd interpretu

Vše začíná třídou *XMLParser* a její metodou *parse*, která provede nad zdrojovým kódem v XML lexikální a syntaktickou analýzu podobně jako ve skriptu *parse.php*. Proměnná *_INST_MAP* mapuje operační kód instrukce

na typy operandů, které přijímá. Zpracované instrukce po analýze kódu předáme třídě *InstructionExecutor*, která řídí hlavní činnost interpretu.

InstructionExecutor si nejprve vytvoří instance všech potřebných tříd a po zavolání metody *interpret* začne vykonávat instrukce v cyklu dokud jsou nějaké na instrukční pásce. Každá instrukce má implementovanou svoji metodu která vykonává její činnost. Tyto metody začínají znakem podtržítka za kterým následuje operační kód instrukce. Díky tomuto formátu je možné volat funkcí *eval* příslušné metody spolu s argumenty instrukce. Sběr a zápis statistik jsem zabudoval do třídy InstructionExecutor, neboť ovládá všechny části interpretu a má přehled o všech sbíraných údajích.

Většina metod pro instrukce jednoduše získá vstupní operandy a provede odpovídající operaci mezi těmito proměnnými, které jsou instancemi třídy *Var*, ta implementuje magické metody pro všechny potřebné operace.

Při implementaci chování interpretu v různých krajních případech či způsobem výpisu výstupu z ladících instrukcí jsem se řídil podle interpretu *ic20int*.

1.1 Rozšíření

V rámci řešení 2. projektu jsem implementoval rozšíření **STATI**, umožňující sběr statistik o vykonávaném kódu, **FLOAT** pro práci s datovým typem float a také **STACK** umožňující používat zásobníkové varianty vybraných instrukcí.

2 Testovaní skript

Skript *test.php* kontroluje přítomnost potřebných souborů a složek s ohledem na zvolené přepínače. Například není potřeba kontrolovat přítomnost interpretu při použití přepínače `--parse-only`.

Pro procházení složek rekurzivně je použita kombinace třídy *RecursiveDirectoryIterator* a dalších iterátorů. Pro procházení jedné složky pak stačí jen funkce *scandir*. Skript si připraví tři dočasné soubory pro zapisování výstupů z testů které následně spustí. Výsledky sbírá do asociativního pole, kde klíči jsou cesty k testům bez přípon. Po spuštění všech testů se sestaví a vypíše na standardní výstup jednoduchá HTML tabulka s barevnými řádky – zelené řádky indikují úspěšný test; červené řádky indikují neúspěšný test.

Testovací skript mi velice pomohl při odhalení chyb které nebyly na první pohled viditelné.