Implementační dokumentace k 2. úloze do IPP 2023/2024

Jméno a příjmení: **Alisher Mazhirinov**
Login: **xmazhi00**

1 Zpracování vstupního souboru

Program dostává od **parse.py** vytvořenou XML reprezentaci pomocí speciálního parametru –**source**. Program předpokládá, že vstupní XML reprezentace neobsahuje lexikální ani syntaktické chyby. Po přijetí souboru se provádí kontrola struktury XML reprezentace a zachování pořadí instrukcí a návěští.

Kontrola vstupní XML reprezentace probíhá v třídě `Interpreter`, která dědí určitou funkcionalitu od abstraktní třídy `AbstractInterpreter`.

Po ověření struktury XML reprezentace se vytváří instance třídy `Instructions`. Poté je pro každou instrukci volána metoda **start_interpreter()**, která spouští samotný interpret. Třída Instructions obsahuje metodu **check_arguments()**, která slouží k ověření počtu argumentů každého operačního kódu.

2 Zpracování instrukcí

Po získání třídou `Instructions` operačního kódu se vytváří instance třídy `Execute`, ve které probíhá zpracování instrukce a provedení všech potřebných akcí pro danou operaci. Postup práce je následující: Na začátku probíhá kontrola, zda je operační kód platný a existuje v `IPPcode24`. Poté následuje kontrola počtu očekávaných argumentů. V závislosti na operačním kódu se provádějí další kontroly typů argumentů, jejich existence v pamětových ramcích a ověření hodnot.

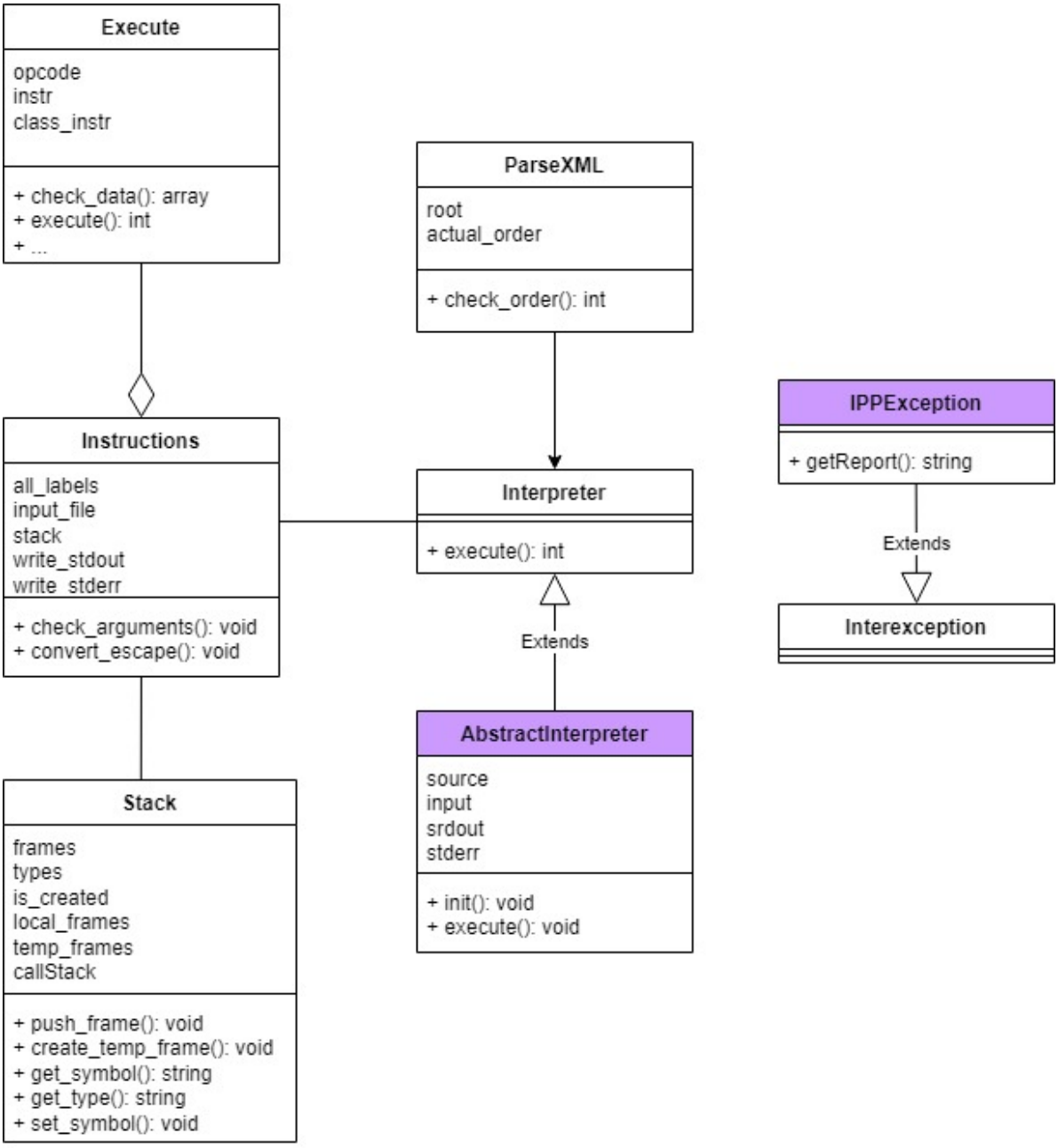
Pro `IPPcode24` existují tři typy paměťových rámců - `GF`, `LF`, `TF`. Práce s nimi probíhá pomocí metod v třídě `Stack`. K vytvoření a uchování rámců se používá metoda **push_frame()** a původní hodnota proměnné je nastavena na hodnotu null. Metody **get_symbol()** a **get_type()** slouží k získání hodnoty a typu proměnné uložené v jednom z paměťových rámců. Metoda **set_symbol()** slouží k modifikaci hodnoty proměnné v jednom z tří paměťových rámců. Metoda **create_temp_frame()** slouží k vytvoření `TF`.

Při výskytu chyb nebo neočekávaného chování interpretátor vyvolává výjimku, která vypíše důvod chyby na standartní chybový výstup a vrátí chybový navratový kód. Tím se zabývá třída `Interexception`, která dědí funkcionalitu od třídy `IPPException`, poskytující možnost vrátit chybu a zastavit program.

3 Zpracování parametrů a spuštění skriptu

Pro spuštění skriptu stačí zadat příkaz **php8.3 ./interpret.php** –**source=sourcefile** –**input=inputfile**. V případě, že chybí argument –**input=inputfile**, předpokládá se, že data pro instrukci READ budou přicházet ze standardního vstupu. Pro výpis nápovědy se používá parametr **--help**, díky kterému skript vypíše krátkou nápovědu a ukončí program.

4 Diagram tříd



Na diagramu je dobře vidět, jak každá třída vzájemně komunikuje. Fialovou barvou jsem označil třídy, které jsou součástí `ipp-core`, což znamená, že nebyly mnou implementovány, ale byly nakonec přidány pro lepší porozumění fungování interpretu a moje implementace.

Rozhodl jsem se také nepřidávat všechny metody třídy `Execute`, které implementují každé operační kódy, pro lepší přehlednost.