Implementační dokumentace k 2. úloze do IPP 2022/2023

Jméno a příjmení: Michal Novák
Login: xnovak3g

18. dubna 2023

1 Úvod

Interpret jazyka IPPcode23 je implementován v jazyce Python na verzi 3.10[1]. Struktura programu je rozdělena mezi 4 moduly. V řešení projektu byla snaha uplatnit objektově orientovaný přístup.

2 Implementované třídy

2.1 Opcode int_libs/resources.py

Atributem je identifikátor operačního kódu.

2.2 Operand int_libs/resources.py

Uchovává informace o argumentech (operandech) instrukce. Díky použití tabulky symbolů (viz níže) jsou shodné argumenty[2] uloženy jako 1 objekt, tzn. při vykonávání instrukcí je možné je použít i pro uchovávání dat v atributu data, není tedy nutné uchovávat data operandů a jejich typy v samostatné tabulce. Změna dat argumetu u 1 instrukce zapříčiní změnu dat tohoto argumentu pro další použití.

2.3 Symtable int_libs/resources.py

Implementuje tabulku symbolů. Do tabulky symbolů se přidávají pouze operandy. Při načtení operandu z XML souboru se nejprve zkontroluje, jestli již tento operand není v tabulce. Pokud není operand nalezen, vytvoří se nový objekt operandu a ten se vloží do tabulky. Tabulka symbolů je pro jednoduchost implementována čistě jako seznam.

2.4 Code int_libs/resources.py

Uchovává informace o celém načteném kódu. Obsahuje tabulku symbolů, jednotlivé instrukce (řádky kódu) a seznam (slovník) definovaných návěští.

2.5 Scanner int_libs/scanner.py

Tato třída slouží k načtení kódu IPPcode23 z XML podoby do vnitřní reprezentace. Metoda `GetTokens` vytvoří a naplní objekt třídy Code získanými daty.
Ke zpracování XML struktury je použita knihovna xml.etree.ElementTree. V průběhu zpracování se kontroluje správnost XML reprezentace. Při objevení chyb v XML se interpretace ukončí s případnými chybovými kódy.

2.6 Parser int_libs/parser.py

V této třídě jsou iplementovány metody pro syntaktickou a sémantickou kontrolu. Kontroluje se postupně existence operačního kódu, definice proměnné, počet argumentů pro zadaný operační kód a dále správný datový typ argumentů. Kontrola datových typů je prováděna porovnáváním s referenčními datovými typy v předem definovaných pravidlech. Vzhledem k povaze jazyka IPPcode23 je nutné provádět kontrolu dynamicky před každým spuštěním instrukce.

[1] Používá specifika pro verzi 3.10 a není tedy zpětně kompatibilní.
[2] Argumenty se stejným identifikátorem a typem.

2.7 Runner int_libs/execution.py

Zde je implementováno samotné vykonávání instrukcí. Vstupem metody ExecuteInstruction je 1 instrukce (řádek kódu), která se má vykonat. Vzhledem k již předchozímu testování se zde ošetřují a kontrolují jen specifické chyby, které by mohly nastat (mimo zásobníkové instrukce, kde se vše kontroluje až při vykonávání instrukce). Pokud se v instrukci provádí skok na návěstí, je návratem této instrukce pozice v kódu, na kterou se má pokračovat. Všechna specifika potřebná pro běh programu se ukládají v atributech této třídy. Jsou zde záznamy o vytvořených rámcích, zásobník pro návraty z funkcí a datový zásobník.

2.8 Interpret interpret.py

Tto třída řídí celý proces implementace. Jejím hlavním úkolem je spojit jednotlivé součásti interpretu. Právě zde je implementováno procházení kódu na základě pořadí instrukcí. Velkou motivací k použitému přístupu byl projekt (Procesor s jednoduchou instrukční sadou) do předmětu INP. Aktuální pozice v kódu se uchovává pomocí programového čítače (program counter). Po vykonání instrukce se čítač inkrementuje do doby, než hodnota čítače odpovídá nějakému pořadí instrukce[3]. V případě skoku v kódu se hodnota v čítači přepíše umístěním, kam se má skok provést.

2.9 Errors int_libs/resources.py

Ukončení běhu interpretu chybovým kódem a hláškou na standardní chybový výstup.

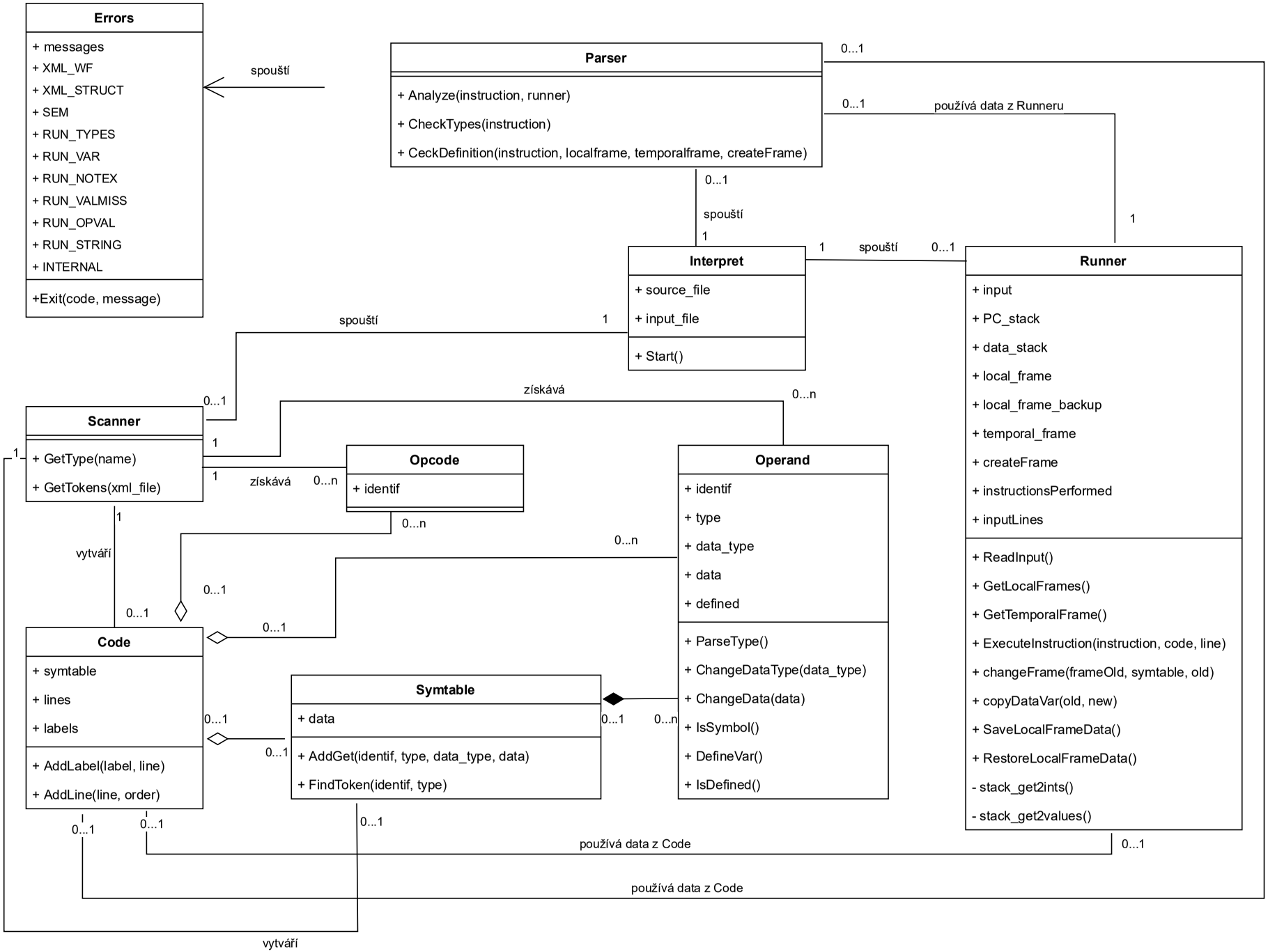
2.10 Types int_libs/resources.py

Výčet možných datových typů.

3 Rozšíření

Součástí řešení projektu je jako jediné z rozšíření implementován STACK. Narozdíl od insrukcí s normálními operndy, kde se sémentické a syntaktické kontroly provádějí před vykonáváním instrukce, jsou kontroly stavu zásobníku prováděny při samotném vykonávání.

[3] Pořadí (order) může být zadáno přerušovaně. Při vykonávání kódu se tyto mezery ignorují.



Obrázek 1: Diagram tříd