Implementační dokumentace k 1. úloze do IPP 2024/2025
Jméno a příjmení: Denys Pylypenko
Login: xpylypd00

Přehled řešení

Skript `parse.py` implementovan pro jazyk SOL25. Ze standardního vstupu přečte zdrojový kód, provede lexikální a syntaktickou analýzu (využívá knihovnu `Lark`), statickou kontrolu (např. existence `Main`, `run`, cyklická dědičnost, redeklarace metod atd.) a výsledný abstraktní syntaktický strom (AST) vypíše v XML formátu. Při chybách (lexikálních, syntaktických, sémantických) skript ukončí běh s příslušným návratovým kódem a chybovou hláškou.

Hlavní komponenty skriptu lze rozdělit do následujících částí

- *Čtení parametrů a vstupního kódu*: Zpracování argumentů skriptu (např. `--help`) a načtení kódu.
- *Definice gramatiky*: Přes `Lark` je vytvořen parser podle gramatiky jazyka SOL25, definované v proměnné `sol25_grammar`.
- *AST a statická analýza*: Třída `ASTTransformer` převádí parse tree na uzly jako `ProgramNode`, `ClassNode`, `BlockNode` atd. Kontrolní funkce (např. `code_check`, `check_block_for_undefined_vars` atd.) dohlížejí na korektnost (zda jsou proměnné před použitím přiřazené, validní odkazy na třídy, správný počet parametrů metod a podobně).
- *Generování XML*: Funkce `build_program_xml` vytváří hlavní element `<program>`, vnořené elementy `<class>` a `<block>` generují rekurzivně strukturu AST, včetně literálů (`<literal>`), proměnných (`<var>`), příkazů (`<assign>`) či volání metod (`<send>`).

Interní reprezentace a hlavní datové struktury

- *ProgramNode*: reprezentuje celý zdrojový kód, drží pole uživatelských tříd.
- *ClassNode*: popisuje deklaraci jedné uživatelské třídy (jméno, rodič, metody).
- *MethodNode*: odpovídá instanční metodě s daným selektorem (např. `run`, `compute:and:`) a tělem (bloku kódu).
- *BlockNode*: modeluje blok kódu se seznamem formálních parametrů a vnitřních příkazů (zatím pouze přiřazení).
- **Node*: pro vyjádření příkazů a výrazů (např. `AssignNode`, `LiteralNode`, `SendNode`) slouží další třídy.

Každý blok (`BlockNode`) uchovává seznam parametrů a seznam přiřazovacích příkazů (instance `AssignNode`). Každý příkaz AssignNode cílí na proměnnou a má výraz, který se může skládat buď z literálu, volání metody (`SendNode`), proměnné (`VarNode`), nebo vnořeného bloku.

Testování

Pro ověření funkčnosti a korektního zachycení chyb (lexikálních, syntaktických i sémantických) bylo otestováno několik programů v jazyce SOL25. Pár z nich:

- Chybějící metoda `run` v hlavní třídě `Main`.
- Cyklická dědičnost: třída `A` dědí `B`, která ale dědí `A`.
- Opakovaná deklarace stejné metody v jedné třídě.
- Program s voláním metody, která má více parametrů, než vyplývá ze selektoru.

Kromě vlastních testů byly pro kontrolu použity také testy od kolehů z Discordu. Ty pomohly odhalit méně zjevné chyby, např. nesprávné zpracování vnořených bloků a volání metod uvnitř nich a další.
Aktuální omezení: Skript má občasné problémy se zpracováním vícenásobného komentáře v řadě a při chybném kódu se někdy nevypisuje hláška úplně podle zadání. Kromě toho nebyly zatím pokryty všechny hraniční případy práce se speciálními znaky.

Výstup a zhodnocení

Výsledkem skriptu je validní XML strom s plnou sémantickou kontrolou. Výstupní XML je formátováno pomocí `minidom` pro přehlednost a správnou strukturu.
Skript využívá standardní knihovny jazyka Python3.11, parser `Lark` a několik dalších povolených. Implementace je rozdělena do jasně definovaných tříd a funkcí, což asi usnadňuje čtení kódu i úpravy. I když základní funkcionalita odpovídá zadání, některé okrajové případy by ještě vyžadovali doladění.