Implementační dokumentace k 1. úloze do IPP 2024/2025

Jméno a příjmení: Jakub Fukala
Login: xfukal01

1. Myšlenka

Skript parse.py je navržen jako lexikální, syntaktický a sémentický analyzátor jazyka SOL25, který transformuje vstupní zdrojový kód do abstraktního syntaktického stromu (AST) a následně generuje XML reprezentaci. Implementace je modulární, s důrazem na objektově orientovaný přístup.

2. Interní reprezentace

- **Gramatika**: Definováné pomocí knihovny Lark s parserem LALR.
- **AST**: Používá hierarchii tříd ProgramNode, ClassNode, MethodNode, BlockNode ,AssignNode, SendNode, LiteralNode, VarNode pro vnitřní reprezentaci definovené v ast_nodes.py.

3. Postup řešení

- **Lexikální analýza**: Implementována pomocí Lark, kde LALR parser zpracovává vstupní zdrojový kód a ověřuje jeho syntaktickou správnost.
- **Syntaktická analýza**: Provádí strukturované zpracování gramatických pravidel, kde se validuje správná kombinace selektorů a parametrů.
- **Převod na AST**: Použití Transformer z knihovny Lark umožňuje přímý převod syntaktického stromu na AST.
- **Semantická analýza**: Provádí kontrolu průchodem AST na nedefinované proměnné, chybně definované třídy, cyklickou dědičnost a nesprávnou aritu selektorů.
- **Vytváření XML**: Finální AST se zpracovává a převádí na XML formát podle definovaných pravidel.

4. OOP a návrhový vzor

V projektu bylo aplikováno rozšíření **NVP (Objektově orientovaný přístup)** a využit návrhový vzor **Návštěvník (Visitor)** pro převod syntaktického stromu na AST:

- Daný navrhový vzor ním umožňuje oddělit logiku transformace od struktury samotného stromu, čímž zajišťuje lepší rozšiřitelnost a přehlednost kódu.
- V souboru parse_to_ast.py, kde Sol25Transformer dědí od Transformer z Lark a postupně zpracovává jednotlivé uzly syntaktického stromu.
- Implementace jednotlivých metod v Sol25Transformer odpovídá jednotlivým uzlům v syntaktickém stromu generovaném Larkem, přičemž vrací odpovídající AST uzly (např. ClassNode, MethodNode). Každá metoda zpracovává konkrétní neterminál definovaný v gramatice a transformuje ho do správné struktury AST, což umožňuje jasnou a přehlednou konverzi.

5. UML diagram

UML diagram ilustruje, jak jsou uzly AST navzájem propojeny. V praxi může expr v AssignNode či arguments v SendNode ukazovat na některou z odvozených tříd (LiteralNode, VarNode, BlockNode apod.).