Implementační dokumentace k 1. úloze do IPP 2021/2022
Jméno a příjmení: David Chocholatý
Login: xchoch09

1 Úvod

Následující dokumentace popisuje implementaci řešení 1. úlohy do předmětu IPP pro akademický rok 2021/2022, která implementuje lexikální a syntaktický analyzátor kódu IPPcode22. V sekci *2.2 Bonusové rozšíření NVP* je také uveden podrobný popis řešení rozšíření pro objektově orientovaný návrh. Úloha byla implementována skripty v jazyce PHP a spouštěna pomocí verze PHP8.1.

2 Implementace

2.1 Postup řešení

Hlavní skript se nachází v souboru `parse.php`. Tento skript obsluhuje případný parametr *–h/––help* pro výpis nápovědy. Dále pracuje se syntaktickým analyzátorem (*třída Parser*) a s metodami vytvářející výstupní XML reprezentaci kódu (*třída Array2Xml* implementovaná v souboru `array_to_xml.php`). Práce s metodami pro XML reprezentací kódu je vytvořena ve funkci *createXml()* ve zmíněném souboru *parse.php*. Syntaktický analyzátor (*třída Parser*) je implementovaný v souboru `syntax_analysis.php`. Tato třída pracuje s lexikálním analyzátorem (*třída Scanner*) a zároveň vytváří pole obsahující strukturu pro převod kódu v IPPcode22 do XML reprezentace. Zmíněný lexikální analyzátor je implementovaný v souboru `lexical_analysis.php`. Tento analyzátor volá metodu pro načtení instrukce (metoda *readInstruction()* vytvořena ve třídě *StringUtil*) a provádí lexikální analýzu implementovanou v metodě *lexicalAnalysis()*. Zmíněná třída *StringUtil* obsahující i další užitečné metody pro práci s řetězci, jako je například mazání komentářů vstupního kódu, je implementována v souboru `string_util.php`. Kromě hlavní struktury programu jsou vytvořeny i další skripty pro dodatečné struktury a metody potřebné pro analyzátor. Mezi tyto struktury patří například datové typy jazyka IPPcode22 (soubor `data_type.php`), typy rámců jazyka (soubor `frame_type.php`) a instrukční sada (soubor `instruction_set.php`). Další struktury implementující vnitřní logiku analyzátoru jsou třída *Instruction* (soubor `instruction.php`) reprezentující instrukci zpracovanou lexikálním analyzátorem, třída *Token* (soubor `token.php`) reprezentující token a zároveň využívající návrhový vzor *Abstraktní továrna* pro bonusové rozšíření *NVP* a třída *TokenUtil* (soubor `token_util.php`) implementující užitečné metody pro práci s tokeny. Program také využívá výčtového datového typu, pro který byla přidána podpora v PHP právě od používané verze PHP8.1. Zmíněný výčtový datový typ je použit pro reprezentaci typu tokenu (soubor `token_type.php`) a pro chybové kódy (soubor `exit_code.php`). Používané definice jsou vytvořeny v posledním doposud nezmíněném souboru `definitions.php`.

2.2 Bonusové rozšíření *NVP*

Při implementaci analyzátoru pomocí objektově orientovaného návrhu byl využit návrhový vzor *Abstraktní továrna*. Tento návrhový vzor je implementovaný v souboru `token.php`. Hlavní třídou návrhu je abstraktní třída *Token* reprezentující token pro analyzátor. Zmíněná třída obsahuje proměnnou `$tokenCode` pro kód tokenu a proměnnou `$tokenVal` pro hodnotu tokenu. Pro získání hodnot zmíněných proměnných jsou implementovány metody *getTokenCode()* a *getTokenVal()*. Poslední metodou třídy je abstraktní metoda *getType()* pro získání typu tokenu, kterou implementují odvozené třídy. Od třídy *Token* dědí třídy *OpCode* reprezentující operační kód, *Operand* reprezentující operand, *EndOfFile* reprezentující konec souboru (vstupního toku) EOF a *LanguageIdentifier* reprezentující identifikátor jazyka. Dále dle použitého návrhového vzoru je vytvořena abstraktní třída TokenFactory reprezentující továrnu tokenů. Tato třída obsahuje abstraktní metodu *createToken()* implementovanou odvozenými třídami. Od třídy *TokenFactory* dědí dále dvě třídy, a to třída *OpCodeFactory* reprezentující továrnu operačních kódů a třída *OperandFactory* reprezentující továrnu operandů. Důvod použití návrhového vzoru *Abstraktní továrna* je především ve využití principu továrny při návrhu. Lexikální analyzátor postupně vytváří tokeny jednotlivých druhů a především velké množství tokenů typu operační kód a operand. Pro tyto dva typy tokenů jsou pro vytváření objektů implementovány továrny. Pak jednotlivé typy tokenů implementují odvozené třídy od třídy *Token*, která je základním stavebním kamenem těchto tříd, ale sama bez dalšího rozšíření neposkytuje všechny potřebné informace pro její samotné praktické využití.