Implementační dokumentace k 1. úloze do IPP 2022/2023
Jméno a příjmení: Dmitrii Ivanushkin
Login: xivanu00

Krátký popis

Výsledkem práce je skript `parse.php`, který pracuje s data ze standardního vchodu a generuje XML reprezentaci programu.

Specifikace řešení

Projekt se skládá z řídícího souboru `parse.php` (obsahuje základní vytváření objektů, ověření argumentů a tisk pomocného textu) a souborů, obsahující pomocné třídy:

- `Argument.php` - Formuje objekt typu argument s položky `typ` a `hodnota`. Soubor také obsahuje abstraktní třídu `ArgumentFactory` která je součásti implementace návrhového vzoru "Tovární metoda" (viz Detaily implementace)
- `Instruction.php` - Formuje objekt typu instrukce s položky `opcode` (operační kód) a `args` (pole argumentů této instrukce). Soubor také obsahuje abstraktní třídu `InstructionFactory` která je součásti implementace návrhového vzoru "Tovární metoda" (viz Detaily implementace)
- `Program.php` - Naplňuje celý výsledek instrukcí a resp. její argumenty. Využívá se při generování výstupu.
- `Line.php` - Pomocí objektu generovaného této třídou program prochází řetězcem a odstraňuje komentáře.
- `Output.php` - Výsledkem je výstup. Objekt pomocí pravidel specifikovaných v zadání tiskne elementy z objektu třídy `Program`

Detaily implementace

Při implementaci projektu jsem použil rozšířeni NVP a vybral jsem návrhový vzor "Tovární metoda" (angl. Factory).

Použil jsem ho při naplněni položek argumentu a instrukce. Vzor funguje tak, že v závislosti na instrukce (roztřídil jsem je do několika skupin dle zadání) volá generátor argumentu, který modifikuje vstupný řetězec, příp. dělí ho a modifikovanou variantou naplní položky objektu třídy `Argument`

Ten generátor jsem se rozhodl dát do jiné abstraktní třídy, která je taky generátorem a volání je přímo ž funkce `main` při inicializaci a ověření instrukce. Takže volám jenom generátor, který bude naplňovat instrukce v závislosti na vstupním řetězci.