Implementační dokumentace k 2. úloze do IPP 2023/2024

Jméno a příjmení: Maksim Samusevich
Login: xsamus00

Úvod

Tento dokument popisuje implementaci interpretu, který načte XML reprezentaci programu a interpretuje ji s využitím vstupů podle parametrů příkazové řádky. Projekt využívá povinně framework ipp-core, který poskytuje základní funkce pro načítání vstupu, výpisy a zpracování parametrů.V implementaci jsou použity návrhové vzory jako je **Factory** pro tvorbu objektů příkazů a **Command** pro reprezentaci jednotlivých instrukcí.

1 Návrhový vzor Command

Návrhový vzor Command je využíván k oddělení výkonné logiky příkazů od jejich vyvolávače. Každý příkaz implementuje rozhraní `Command`, které obsahuje metodu `execute()`. Tato metoda je pak volána interpretem, což odděluje logiku vykonávání příkazu od jeho vyvolání. V našem projektu je tento vzor aplikován na různé příkazy, jako jsou `ArithmeticCommand`, `LogicCommand`, a další. Každá třída příkazu specifikuje své chování v metode `execute()`, které se dynamicky vyvolává za běhu programu, když interpreter zpracovává jednotlivé instrukce.

2 Návrhový vzor Factory

Návrhový vzor Factory je použit pro vytváření objektů příkazů na základě typu operace, kterou je třeba vykonat. `CommandFactory` je centrální komponenta, která analyzuje instrukce získané z parseru a na jejich základě vytváří specifické objekty příkazů. Tento proces dekomponuje vytváření objektů od jejich použití, což umožňuje flexibilně reagovat na změny ve specifikaci příkazů bez nutnosti přepisovat kód, který tyto příkazy vyvolává. Factory vzor zde také zajišťuje, že všechny příkazy jsou správně inicializovány s potřebnými závislostmi, což znamená, že každý příkaz dostane přístup k potřebným zdrojům, jako jsou vstupní a výstupní rozhraní, či kontext programu.

3 Třídy a funkce:

Interpreter

Účel: Třída `Interpreter` slouží jako vstupní bod interpretace IPPcode24, který řídí celkový běh programu od načtení XML dokumentu až po vykonání instrukcí.

Metody:

- `execute()`: Hlavní metoda, která spouští celý proces interpretace. Načítá XML dokument, parsuje ho a následně spustí interpretaci instrukcí v cyklu.

XmlParser

Účel: `XmlParser` je statická třída zodpovědná za parsování vstupního XML dokumentu a jeho transformaci na strukturovanou formu vhodnou pro další zpracování.

Metody:

- `parse(\DOMDocument $domDocument)`: Analyzuje strukturu `DOMDocument`, extrahuje z něj jednotlivé instrukce a jejich argumenty a vrátí strukturovaný seznam instrukcí.
- `stringEscape(string $string)`: Upravuje řetězce ve vstupních datech tak, aby byly korektně zpracovatelné, převádí escape sekvence na odpovídající znaky.
- `sortInstructionsByOrder()`: Řadí instrukce podle atributu 'order'.

CommandFactory

Účel: Třída `CommandFactory` slouží k dynamickému vytváření specifických příkazů na základě operací (opcodes) definovaných v IPPcode24.

Metody:

- `createCommand(mixed $instruction, Program $program)`: Factory metoda pro vytváření příkazů. Na základě operace (opcode) vytváří příslušný objekt command.

Program

Účel: Třída `Program` slouží jako kontejner a správce všech vykonávaných instrukcí a proměnných v průběhu interpretace.

Metody:

- `run()`: Řídí vykonávání načtených instrukcí, které jsou uchovávány v instanci třídy.
- `_getLabels()`, `_check()`, `_arithmetic()`, `_strings()`, `_logic()`, `findLabelIndex()`: Tyto metody poskytují implementaci specifických funkcionalit, jako je práce s proměnnými, vykonávání aritmetických a logických operací, atd.

ArithmeticCommand, StringOperationCommand, LogicCommand (a další command třídy)

Účel: Tyto třídy implementují rozhraní Command a jsou zodpovědné za vykonání specifických skupin instrukcí.

Metody:

- `execute()`: Vykonává logiku specifickou pro daný typ operace. Tato metoda čte vstupní argumenty z `Program`, aplikuje na ně logiku instrukce a uloží výsledky zpět do `Program`.

AbstractInterpreter

Popis: Třída `AbstractInterpreter` je abstraktní základní třída, poskytující základní strukturu pro interprety. V rámci našeho projektu poskytuje přístup k základním I/O objektům.

Využití:

- `source.getDOMDocument()`: Načítá DOM dokument, což je první krok v procesu interpretace.
- `input.readString()`: Slouží pro načítání vstupních dat od uživatele.
- `stdout.writeString()` a `stderr.writeString()`: Používány pro výstup textových dat a chybových hlášení.

I/O Interfaces

Popis: Rozhraní `InputReader` a `OutputWriter` jsou klíčové pro správu vstupů a výstupů.

Využití:

- `InputReader` (`input`): Čte data zadávaná uživatelem.
- `OutputWriter` (`stdout`, `stderr`): Slouží pro zápis výstupů a chybových zpráv.

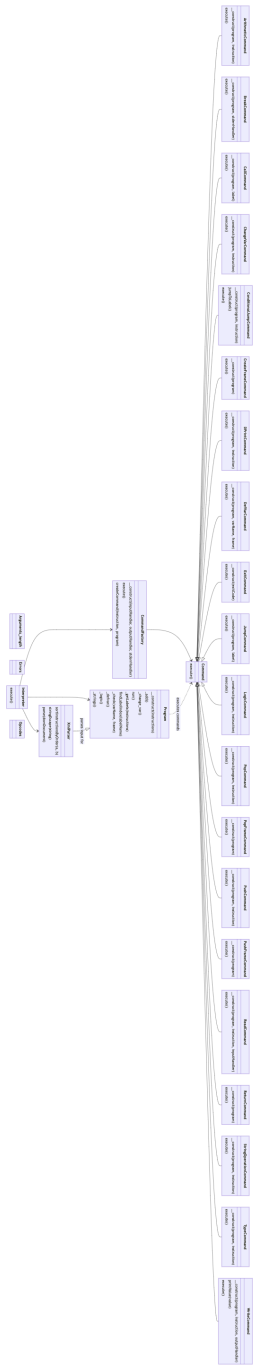

Figure 1: UML diagram