Implementační dokumentace k 2. úloze do IPP 2021/2022

Jméno a příjmení: Vilém Gottwald
Login: xgottw07

Interpret XML reprezentace kódu

Zpracování argumentů příkazové řádky

Za účelem zpracování argumentů příkazové řádky byla vytvořena třída `ArgumentProcessor`. Při její instanciaci se argumenty příkazové řádky postupně projdou a případně uloží do instančních atributů nově vzniklého objektu. Parametry, které nebyly nalezeny jsou reprezentovány objektem `None`. V případě nesprávné kombinace argumentů příkazové řádky je skript ukončen s odpovídajícím návratovým kódem (viz třída `ReturnValues`).

Interní reprezentace programu

Program je reprezentován objektem třídy `Program`, která obsahuje seznam objektů reprezentující instrukce. Každá instrukce je instancí stejnojmenné třídy, přičemž konstruktor všech instrukcí je děděn z rodičovské třídy `Instruction`. Program dále obsahuje reference na objekt třídy `Stack`, která reprezentuje datový zásobník programu a programový čítač reprezentovaný objektem třídy `ProgramCounter`. Jednotlivé rámce programu jsou reprezentovány speciálními třídami, ale jejich společné metody jsou implementovány v rodičovské třídě `Frame`. Třídy reprezentující jednotlivé datové typy se nachází v modulu `data_types`.

Zpracování XML reprezentace kódu

Zpracování XML reprezentace kódu probíhá ve třídě `Program` voláním metody `load`. Tato metoda za použití modulu `xml.etree.ElementTree` zpracuje vstupní XML a vytvoří seznam obsahující instance jednotlivých instrukcí. Následně jsou instrukce na základě jejich atributu určujícího pořadí seřazeny a je vytvořen slovník obsahující názvy návěští a jejich indexy v seznamu instrukcí.

Interpretace programu

Interpretace zdrojového programu je vyvolána metodou `interpret` třídy `Program`. Tato metoda postupně projde všechny instrukce v seznamu instrukcí a zavolá jejich metody `exec`, které způsobí vykonání dané instrukce.

Statistiky interpretace

Vytváření statistik bonusového rozšíření STATI zajišťuje třída `Stats`. Statistiky jsou vytvářeny voláním metody `countIt`, které je argumentem předána právě prováděná instrukce. Na konci programu jsou vypočtené statistiky vypsány na standardní výstup pomocí metody `printStats`.

Testovací rámec

Zpracování argumentů příkazové řádky

Zpracováním argumentů příkazové řádky se zabývá třída `CLA_GetSettings`. Jednotlivé argumenty jsou uloženy jako instanční atributy a mají přednastavené výchozí hodnoty. Při nalezení validních argumentů tedy dochází pouze k aktualizaci jejich hodnot. V případě nesprávné kombinace argumentů příkazové řádky je skript ukončen s odpovídajícím návratovým kódem (viz třída `ReturnValues`).

Nalezení souborů s testy

K vyhledání souborů s testy slouží statická metoda `getTestsFromDir` ze třídy `TestFilesManager`, která rekurzivně projde zvolený adresář a vrátí seznam cest k hlavním souborům testů. Následně dochází u každého testu ke kontrole existence všech jeho podružných souborů a případnému vytvoření chybějících.

Testování

Provedení testů je implementováno ve třídě `Tester`, přičemž způsob testování je vždy zvolen na základě konfigurace skriptu při spuštění. Při samotném testování se nejdříve ze souboru načte očekávaný návratový kód testu a na základě jeho hodnoty dojde ke spuštění testovaného skriptu. V případě, že se jedná o kód značící neúspěch, je uložen a vyhodnocen pouze návratový kód a výstup testovaného skriptu je zahozen. V opačném případě je uložen i výstup, který je následně porovnán s referenčním výstupem pomocí nástroje diff, nebo A7Soft JExamXML (v případě testování analyzátoru). Výsledky testů (objekty třídy `TestResult`) jsou po každém testu přidávány do seznamu výsledků testů uloženém v instančním atributu.

Výstupní HTML

Tvorba výstupního HTML je řešena ve třídě `HtmlWriter`. Tato třída používá úryvky HTML kódu uložené v modulu `HtmlSnippets`, s jejichž pomocí na standardní výstup vypisuje soubor s výsledky testování. Při tvorbě výsledků jednotlivých testů je seznam výsledků testů postupně procházen a HTML je vypisováno pro každý test zvlášť.