Programowanie obiektowe zadanie 2

Wersja 1.01

Celem zadania będzie stworzenie interpretera pewnego prostego języka programowania o nazwie ROBSON.

Twój program powinien umożliwiać:

- Wczytanie programu w języku ROBSON zapisanego w formacie JSON
- Uruchomienie programu w języku ROBSON
- Konwersję programu w języku ROBSON do kompilowalnego programu w Javie
- Zapis wczytanego programu do pliku w formacie JSON.

W języku ROBSON Program w składa się z następujących elementów:

- Instrukcji Bloku wykonanie bloku instrukcji polega na wywołaniu funkcji wykonaj dla każdej instrukcji z bloku.
- wyrażenia logiczne stałe prawda fałsz operacje Oraz Not Lub a także operacje porównujące wyrażenie arytmetyczne większe mniejsze równe
- wyrażenia arytmetyczne stała liczbowa Plus Minus Mnoz dziel
- instrukcję warunkową if składającej się z wyrazanienia logicznego oraz bloku instrukcji wykonywanej w przypadku gdy obliczenie wyrażenia da wartość true oraz bloku instrukcji wykonywanej w przeciwnym przypadku
- Instrukcja pętli składa się wyrażenia logicznego oraz bloku instrukcji
  Wykonanie Instrukcji pętli polega cyklicznym wykonywaniu następujących operacji obliczenia wartości wyrażenia logicznego jeśli to obliczenie da wartość True to wykonujemy blok instrukcji, w przeciwnym wypadku kończymy wykonanie instrukcji
- W programie dostępne są zmienne globalne (początkowo zainicjalizowane na 0) i instrukcje przypisania.

Programy z naszym języku będą zapisywane w piłkach JSON.

Wymagania techniczne

- Główna klasa rozwiązania powinna mieć nazwę `Robson`
- Klasa `Robson` powinna udostępniać następujące metody
  - `void fromJSON(String filename)throws NieprawidlowyProgram` wczytanie programu z pliku JSON
    (wyjątek jest zadeklarowany na potrzeby przyszłych rozszerzeń, w zadaniu możesz założyć, że programy będą poprawne składniowo)
  - `void toJSON(String filename)` zapisanie programu do pliku JSON
  - `void toJava(String filename)` zapisanie programu do pliku Java

    - `double wykonaj() throws BladWykonania` wykonanie programu i zwrócenie wartości liczbowej (lub wyjątku w przypadku błędu)
- Przy zapisie programu do języka Java, należy zapewnić żeby wynikowy program się kompilował i wypisywał na standardowe wyjście wynik wyrażenia które zawiera
- Należy także dostarczyć przykładowy program testujący działanie języka ROBSON:
    - program może być dostarczony jako JUnitTest
    - albo własna klasa testująca
    - może to być np przykład implementacja algorytmu Euklidesa w języku Robson lub program sterujący robami w niestandardowy sposób w przypadku implementacji zadania bonusowego
- Program powinien zawierać kompletną instrukcję kompilacji i uruchomienia (łącznie z podaniem wszystkich niezbędnych bibliotek). Innym sposobem jest przygotowanie projektu Maven zawierającego wszystkie reguły i zależności.

Instrukcje

W formacie JSON każda instrukcja będzie odpowiadać słownikowi, który będzie zawierał obowiązkowe pole “typ” oraz dodatkowe argumenty wg następującej tabeli

Typ	Dodatkowe atrybuty
`Blok`	`instrukcje` - lista instrukcji lub wyrażeń
`If`	`warunek` - wyrażenie opisujące warunek logiczny `blok_prawda` - wyrażenie które ma być wyliczone gdy warunek jest prawdziwy `blok_falsz` - (atrybut opcjonalny) wyrażenie które ma być wykonane gdy warunek jest fałszywy
`While`	`warunek` - wyrażenie opisujące warunek logiczny `blok` - wyrażenie które ma być wykonane w pojedynczej iteracji pętli
`Przypisanie`	`nazwa` - nazwa zmiennej do przypisania (str) `wartosc` - wyrażenie opisujące prawą stronę przypisania
`Plus|Minus|Razy|Dzielenie`	`argument1` - wyrażenie opisujące pierwszy argument `argument2` - wyrażenie opisujące drugi argument
`And|Or`	`argument1` - wyrażenie opisujące pierwszy argument `argument2` - wyrażenie opisujące drugi argument
`< | > | <= | >= | ==`	`argument1` - wyrażenie opisujące pierwszy argument `argument2` - wyrażenie opisujące drugi argument
`Not`	`argument` - wyrażenie opisujące argument

`Liczba`	`wartosc` - wartość stałej liczbowej (double)
`True|False`	-
`Zmienna`	`nazwa` - nazwa zmiennej (str)

Każda instrukcja wylicza wartość wyniką używając następujących reguł

Typ	**Wartość wynikowa**
`Blok`	Wartość ostatniej instrukcji lub 0 w przypadku pustego bloku
`If`	Wartość blok_prawda jeśli warunek jest prawdziwy lub wartość blok_falsz wpp
`While`	0
`Przypisanie`	Wartość zmiennej (po przypisaniu)
`Plus|Minus|Razy|Dzielenie`	Wartość wyniku wykonania operacji (liczbowa)
`And|Or`	Wartość wyniku wykonania operacji (0 - fałsz, 1 - true)
`< | > | <= | >= | ==`	Wartość wyniku wykonania operacji (0 - fałsz, 1 - true)
`Not`	Wartość wyniku wykonania operacji (0 - fałsz, 1 - true)
`Liczba`	Wartość liczby
`True|False`	0 - fałsz, 1 - true
`Zmienna`	Wartość zmiennej

Przykład programu w formacie JSON

Dla programu:

```
{
   "typ":"Blok",
   "instrukcje":[
      {
      "typ": "Plus",
      "argument1":{
            "wartosc": 7.0,
            "typ": "Liczba"
      },
```

```
      "argument2":{
            "wartosc": 8.0,
            "typ": "Liczba"
      }
      }
   ]
}
```

Oczekiwaną wartością wyrażenia powinno być 15.

Przykład 2
Ciąg Fibonacciego:

```
{
      "typ":"Blok",
      "instrukcje":[
      {
            "typ":"Przypisanie",
            "nazwa":"numer",
            "wartosc":{
                  "typ":"Liczba",
                  "wartosc":10
            }
      },
      {
            "typ":"If",
            "warunek":{
                  "typ":"<=",
                  "argument1":{
                        "typ":"Zmienna",
                        "nazwa":"numer"
                  },
                  "argument2":{
                        "typ":"Liczba",
                        "wartosc":2
                  }
            },
            "blok_prawda":{
                  "typ":"Blok",
                  "instrukcje":[
                        {
                        "typ":"Liczba",
                        "wartosc":1
                        }
                  ]
            },
```

```
"blok_falsz":{
      "typ":"Blok",
      "instrukcje":[
            {
            "typ":"Przypisanie",
            "nazwa":"x1",
            "wartosc":{
                  "typ":"Liczba",
                  "wartosc":1
            }
            },
            {
            "typ":"Przypisanie",
            "nazwa":"x2",
            "wartosc":{
                  "typ":"Liczba",
                  "wartosc":1
            }
            },
            {
            "typ":"Przypisanie",
            "nazwa":"index",
            "wartosc":{
                  "typ":"Liczba",
                  "wartosc":3
            }
            },
            {
            "typ":"While",
            "warunek":{
                  "typ":"<=",
                  "argument1":{
                        "typ":"Zmienna",
                        "nazwa":"index"
                  },
                  "argument2":{
                        "typ":"Zmienna",
                        "nazwa":"numer"
                  }
            },
            "blok":{
                  "typ":"Blok",
                  "instrukcje":[
                        {
                              "typ":"Przypisanie",
                              "nazwa":"temp",
                              "wartosc":{
                              "typ":"Plus",
```

```
                    "argument1":{
                        "typ":"Zmienna",
                        "nazwa":"x1"
                    },
                    "argument1":{
                        "typ":"Zmienna",
                        "nazwa":"x2"
                    }
                    }
                },
                {
                    "typ":"Przypisanie",
                    "nazwa":"x1",
                    "wartosc":{
                    "typ":"Zmienna",
                    "nazwa":"x2"
                    }
                },
                {
                    "typ":"Przypisanie",
                    "nazwa":"x2",
                    "wartosc":{
                    "typ":"Zmienna",
                    "nazwa":"temp"
                    }
                },
                {
                    "typ":"Przypisanie",
                    "nazwa":"index",
                    "wartosc":{
                    "typ":"Plus",
                    "argument1":{
                        "typ":"Zmienna",
                        "nazwa":"index"
                    },
                    "argument2":{
                        "typ":"Liczba",
                        "wartosc":1
                    }
                    }
                }
            ]
        }
        },
        {
        "typ":"Zmienna",
        "nazwa":"x2"
        }
```

```
                    ]
                }
            }
            ]
}
```

Ocenianie

- wykonywania programu: do 7 punktów
- export i import programu Javy do plików JSON: 1 punkty
- export wczytanego programu do kodu w Javie: 2 punkty

Przewidziany jest także bonus 2 punkty za implementacje rozszerzenia języka o instrukcje sterujące Robem z zadania 1 Bonus działa w ten sposób że punkty z niego uzyskane uzupełniają ewentualnie utracone punkty

Wskazówki

- Format JSON:
    - https://en.wikipedia.org/wiki/JSON
- Przykładowe biblioteki do parsowania formatu JSON w Javie:
    - https://cliftonlabs.github.io/json-simple/
        - Krótki tutorial: https://mkyong.com/java/json-simple-how-to-parse-json/
    - https://github.com/square/moshi
    - https://github.com/google/gson
        - Bardzo dobry tutorial dla Gson https://github.com/google/gson/blob/master/UserGuide.md