TKOM - dokumentacja wstępna projektu Currex

Kamil Strójwąs

Marzec 2024

1 Opis wstępny

Celem projektu jest stworzenie intepretera języka umożliwiającego wykonywanie operacji walutowych na różnych walutach jednocześnie. W języku tym występują typy znane z innych języków takie jak int, float, string i bool. Ponadto, w języku tym występuje typ walutowy, będący reprezentacją kwoty pieniężnej w danej walucie. Interpreter jest stworzony w języku Java.

2 Specyfikacja języka

W języku istnieją następujące silnie typowane, statyczne typy wbudowane:

- int reprezentujący typ całkowitoliczbowy,
- float będący typem zmiennoprzecinkowym,
- bool czyli typ reprezentujący wartości "prawda"lub "fałsz",
- string będący ciągiem znaków,
- currency będący typem złożonym

Typ cureency składa się z następujących pól:

- name - trójznakowa nazwa waluty, możliwe wartości są jedynie takie które znajdują się w pliku konfiguracyjnym
- value - wartość tej waluty wyrażona w typie stałoprzecinkowym

Dla typu currency domyślnie możliwe są operacja dodawania, odejmowania i przyrównywania.

Dodawanie i odejmowanie jest możliwe dla dwóch dowolnych zmiennych typu currency, nawet jeżeli reprezentują one dwie różne waluty. W takim przypadku, zmienna znajdująca się po prawej stronie operatora zostaje przekonwertowana do tej samej waluty która jest reprezentowana przez zmienną po lewej stronie tego operatora. Konwersja następuje zgodnie z przelicznikiem znajdującym się w tabeli z pliku konfiguracyjnego.

Operator przyrównania '==' dla typu currency zwraca boola. Dla przypadku w którym obie zmienne mają taką samą wartość w polu "name"i taką samą wartość w polu "value", przyrównanie zwróci wartość "true". W każdym innym przypadku otrzymamy "false". Operatory przyrównania '<=', '<', '>=' i '>' mogą zostać wykorzystane dla typu walutowego tylko dla zmiennych o tej samej wartości w polu "name"po czym następuje porównanie wartości w polu "value", w innym przypadku zgłoszony zostanie błąd.

Możliwe jest równiez przelewanie pieniędzy z jednej zmiennej typu currency do innej. Odbywa się to za pomocą operatora ->, gdzie strzałka wskazuje kierunek w którym odbywa się przelew.

Możliwe jest również rzutowanie wartości waluty na inną walutę za pomocą operatora rzutowania '@'. Nie zmienia on wartości waluty którą chcemy rzutować. Możliwa jest też konwersja zmiennej typu walutowego

z jednej waluty w inną wykorzystując jego przelicznik za pomocą operatora '->'. Oba operatory mają lewostronną łączność.

Każda linijka zakończona jest znakiem średnika ';'. Bloki kodu odzielone są klamrami '{}'. Komentarze zaczynają się od podwójnych forward slashy '//'. Do pwypisania czegoś do konsoli stosowana jest funkcja wbudowana print(). Do pobrania wartości od użytkownika służy funkcja input(). Wartości zmiennych są kopiowane przy przypisaniu i podaniu jako argument wywołania funkcji.

Priorytety operatorów wyglądają następująco, 1 to najwyższy priorytet, 9 to najniższy:

Priorytet	Operator	Symbol
1	Dostęp do pola	'.'
2	Operatory unarne	'-', '!'
3	Operatory rzutowania	'@', '->'
4	Mnożenie i dzielenie	'*', '/'
5	Dodawanie i odejmowanie	'+', '-'
6	Przyrównanie	'==', '!=', '<=', '<', '>', '>='
7	AND	'&&'
8	OR	'||'
9	Przypisanie	'='

3 Gramatyka

3.1 Pliku konfiguracyjnego

```
tabela             = waluty, {rząd_konwersji};
rząd_konwersji     = identyfikator, {przelicznik}, ";";
waluty             = {identyfikator}, ";";
przelicznik        = float;
```

3.2 Języka

```
program            = {deklaracja_funkcji};
blok               = "{", {instrukcja}, "}";
instrukcja         = (przypisanie | return), ";" | wyrażenie_if | wyrażenie_while;

przypisanie        = wyrażenie_dostępu, [operator_przypisu, wyrażenie];
wyrażenie_while    = "while", "(", wyrażenie, ")", blok;
wyrażenie_if       = "if", "(", wyrażenie, ")", blok, {"else", ["if", "(", wyrażenie, ")"], blok};
return             = "return ", [wyrażenie];
print              = "print", "(", wyrażenie, ")";
input              = "input", literał;
parseCurrency      = "parseCurrency", "(", identyfikator ,")";

wyrażenie          = wyrażenie_or;
wyrażenie_or       = wyrażenie_and, {operator_or, wyrażenie_and};
wyrażenie_and      = porównanie, {operator_and, porównanie};
porównanie         = wyrażenie_dodania, [przyrównanie, wyrażenie_dodania];
wyrażenie_dodania  = wyrażenie_mnożenia, {operator_addytywny, wyrażenie_mnożenia};
wyrażenie_mnożenia = wyrażenie_rzutu, {multiplikacje, wyrażenie_rzutu};
wyrażenie_rzutu    = wyrażenie_unarne, {(operator_rzutu | operator_wymiany), nazwa_waluty}
```

```
wyrażenie_unarne   = [operator_unarny], (wyrażenie_dostępu | literał);
wyrażenie_dostępu  = podst_wyrażenie, {operator_dostępu, iden_lub_wywołanie};
podst_wyrażenie    = iden_lub_wywołanie | literał_walutowy | "(", wyrażenie ")";

deklaracja_funkcji = typ, nazwa_funkcji, "(", [lista_parametrów], ")", blok;
iden_lub_wywołanie = identyfikator, ["(", [lista_argumentów], ")"];
lista_parametrów   = parametr_funkcji, {",", parametr_funkcji};
parametr_funkcji   = typ, identyfikator;
lista_argumentów   = wyrażenie, {",", wyrażenie};
nazwa_funkcji      = identyfikator;

operator_dostepu   = ".";
operator_unarny    = "-" | "!";
multiplikacje      = "*" | "/";
operator_addytywny = "+" | "-";
przyrównanie       = "<" | "<=" | ">=" | ">" | "==" | "!=";
operator_and       = "&&";
operator_or        = "||";
operator_przypisu  = "=";
operator_rzutu     = "@";
operator_wymiany   = "->";

literał            = boolean | liczba | float | string;
typ                = "int" | "float" | "string" | "bool" | "currency";
identyfikator      = litera, {"_" | litera | cyfra};

literał_walutowy   = [" ", nazwa_waluty];
nazwa_waluty       = {litera}
string             = """, {znak}, """;
float              = liczba, ".", {cyfra};
liczba             = "0" | (["-"], (cyfra_bez_zera, {cyfra}));
boolean            = "true" | "false";

średnik            = ";";
znak               = cyfra | litera | symbol;
symbol             = " " | "," | "." | "!" | "?" | ":" | "-" | "_" | "/";
litera             = "a" | "b" | ... | "z" | "A" | "B" | ... | "Z";
cyfra              = "0" | cyfra_bez_zera;
cyfra_bez_zera     = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9";
```

4 Sposób uruchomienia

Do uruchomienia wymagany jest plik konfiguracyjny zawierający konwersje walut w postaci tabelarycznej. Przykład takiego pliku wygląda następująco:

```
EUR USD PLN;
EUR 1 1.08 4.68;
USD 0.92 1 4.31;
POL 0.18 0.23 1;
```

Do uruchomienia programu należy użyć następującej komendy:

```
java Currex <ścieżka do pliku konfiguracyjnego>
```

5 Konstrukcje języka

W języku, dostępne jest kilka wbudowanych konstrukcji, takich jak pętle, wyrażenia porównania lub rzutowanie walut.

5.1 Operacje arytmetyczne

Język zezwala na podstawowe operacje arytmetyczne na zmiennych typu całkowitego, zmiennoprzecinkowego i walutowego.

```
int one = 1;
int two = 2;
int three = one + two;
// three = 3

int diff = one - two;
// diff = -1

float half = 0.5;
float one_and_half = 1.5;
float multiplication = half * one_and_half;
// multiplication = 0.75
float division = one_and_half / half;
// one_and_half = 3.0
```

5.2 Dostępne operacje na typie walutowym

Z typem walutowym możemy wykonywać podstawowe operacje arytmetyczne takie jak dodawanie, odejmowanie, mnożenie i dzielenie. Wynikiem takiej operacji zawsze będzie typ walutowy. Typ ten zezwala także na rzutowanie do innej waluty bez zmiany nominału, lub przekonwertowanie jej do innej przy użyciu licznika.

```
10.45 PLN + 5 = 15.45 PLN;
10.45 PLN - 15 = -4.55 PLN;
10.45 PLN * 10.5 = 109.725 PLN;
10.45 PLN / 3.5 = 2.98571428571 PLN;
10.45 PLN @ ABC = 10.45 ABC;
// zakładając przelicznik PLN na ABC równy 1 PLN = 2.00 ABC
10.45 PLN -> ABC = 20.90 ABC;
```

Aby zachować jak najlepszą precyzję, typ walutowy jest zaokrąglany do 10 miejsc po przecinku.

5.3 Operacje na typie string

Typ string pozwala jedynie na konkatenację łańcuchów znaków za pomocą operatora dodawania.

```
string str1 = "Projekt "
string str2 = "wstępny"
string str3 = str1 + str2;
// str3 = "Projekt wstępny"
```

5.4 Porównania i instrukcje warunkowe

W języku dostępne są porównania wartości dwóch zmiennych. Możliwe jest także wykorzystanie dwóch operatorów logicznych AND (&&) lub OR (||) aby połączyć kilka warunków w jeden. Instrukcje warunkowe tworzone są za pomocą słów kluczowych "if"oraz "else", które można razem łączyć do postaci "else if"aby stworzyć bardziej zaawansowane warunki.

```
bool isOk = true;
int value1 = 0;
int value2 = 1;

if (value == 1) {
  isOk = false;
}
else if (value == 0 && value2 >= 5) {
  isOk = true;
}
else if (value != 0 || value2 < 0) {
  isOk = true;
}
else {
  isOk = false;
}
// isOk = false
```

5.5 Wyświetlanie w konsoli

W języku dostępne jest wyświetlanie na wyjście komunikatów w konsoli dla użytkownika za pomocą słowa kluczowego "print".

```
int one = 1;
print(one);
// 1

int ten = 10;
print(one + ten);
// 11

print(ten >= one);
// true

currency euros = 2.50 EUR;
currency dollars = 1.00 USD;
print(euros + dollars @ EUR);
// 3.50 EUR
```

5.6 Funkcje

W języku możliwe jest definiowanie swoich własnych funkcji. Mogą one zwracać jakąś wartość, ale jest także dozwolone definiowanie funkcji nie zwracających niczego.

```
currency first = 1.00 ABC;

currency addTen(currency value) {
  currency result = value + 10.00;
  result = result @ XYZ;
  return result;
}

currency final = addTen(first);
// final = 11.00 XYZ
```

```
showValue(currency value) {
  currency result = value @ "ABC";
  print(result);
}

showValue(final);
// 11.00 ABC
```

5.7 Pętle

W języku dostępna jest jedna pętla tworzona przy pomocy słowa kluczego "while".

```
int counter = 0;

while (counter < 5) {
  print(counter);
  print("\n");
  counter = counter + 1;
}
// 0
// 1
// 2
// 3
// 4
```

6 Przykłady kodu

6.1 Możliwe operacje na typie currency

```
currency CreateCurrencyValue() {
    // stworzenie zmiennej typu currency
    currency curr = 34.53 PLN;

    // odwołanie się do pola value
    print(curr.value);
    // 34.53

    // dodatnie jednej złotówki
    curr += 1.00;
    // curr = 35.53 PLN

    // konwersja waluty do innej zakładając kurs 1 PLN = 0.23 USD
    curr = curr -> USD;
    // curr = 8.17 USD

    // rzutowanie waluty na inną walutę z zachowaniem nominału
    // bez faktycznej zmiany wartości zmiennej
    curr_copy = curr @ PLN + 5.00;
    // curr = 8.17 USD
    return curr_copy;
    // 13.17 PLN
}
```

6.2 Operacje arytmetyczne na typie currency

```
showCurrencyOperations() {
    currency curr1 = 1.30 EUR;
    currency curr2 = 1.30 EUR;

    // dodanie do siebie wartości dwóch zmiennych będacymi
    // tymi samymi walutami
    currSum = curr1 + curr2;
    // currSum = 2.30 EUR

    // odjęcie od siebie wartości dwóch zmiennych będacymi
    // tymi samymi walutami
    currDiff = curr1 - curr2;
    // currDiff = 0.30 EUR

    // dodanie do siebie wartości dwóch zmiennych będacymi
    // różnymi walutami
    // zakładamy kurs 1 EUR = 4.68 PLN
    curr1 = 1.00 EUR;
    curr1 = curr1 -> PLN;
    currSum = curr1 + curr2;
    // currSum = 5.98 PLN

    // odjęcie od siebie wartości dwóch zmiennych będacymi
    // różnymi walutami
    currDiff = curr1 - curr2;
    // currDiff = -3.38 PLN
}
```

6.3 Instrukcje warunkowe

```
compare_currency_values(currency curr1, currency curr2) {
    if (curr1 == curr2) {
        print("equal values");
    }
    else {
        print("not equal values");
    }

    if (curr1 > curr2) {
        print("curr1 is bigger than curr2");
    }
return;
}

compare_currency_values(1.30 EUR, 1.30 EUR);
// equal values
```

6.4 Pobranie od użytkownika wartości currency i sparsowanie jej

```
currency getCurrency() {
    string user_input = input("Podaj poprawną wartość currency: ");
    // Podaj poprawną wartość currency: 39 EUR
    currency parsed = parseCurrency(user_input);
    parsed += 10.50;
    return parsed;
    // 49.50 EUR
}
```

6.5 Rzutowanie waluty za pomocą operatora rzutowania

```
currency changeCurr() {
currency curr1 = 5.00 EUR;
currency curr2 = 10.00 USD;
// rzutowanie wartości typu walutowego z jednej waluty w inną
// bez zmiany nominału
curr2 = curr1 @ USD + curr2;
// curr2 = 15.00 USD
// rzutowanie wartości typu walutowego z jednej waluty na
// walutę taką jak w innej zmiennej
return curr2 @ curr1.name;
// 15.00 EUR
}
```

6.6 Zmiana nazwy waluty

```
currency changeName() {
currency curr = 5.00 EUR;
curr.name = PLN;
return curr;
// curr = 5.00 PLN;
}
```

7 Testowanie

Poprawność implementacji języka zostanie sprawdzona z użyciem testów jednostkowych w metodyce TDD.