Zadanie 3 27, 30 października 2025 r.

kurs języka Java

wyrażenia arytmetyczne

Instytut Informatyki
Uniwersytetu Wrocławskiego Paweł Rzechonek

Zaprogramuj i przetestuj hierarchię klas, dzięki której będzie można zbudować drzewo wyrażenia arytmetycznego (liście to operandy a węzły wewnętrzne to operatory arytmetyczne lub funkcje matematyczne). Drzewo wyrażenia można do używać obliczania wartości tego wyrażenia dla przygotowanych wcześniej zmiennych występujących w tym wyrażeniu.

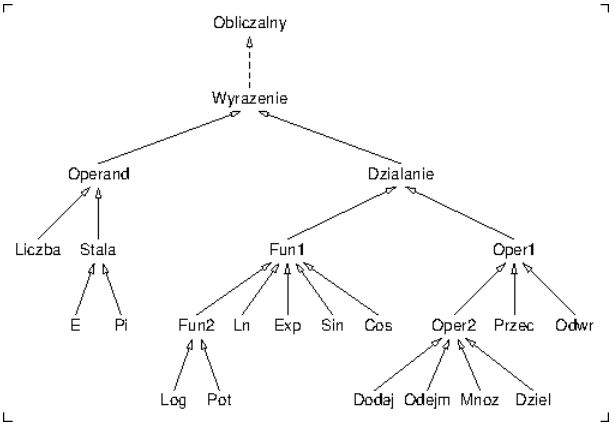
Zdefiniuj interfejs `Obliczalny`, reprezentujący obiekty, na których można coś policzyć metodą `oblicz()`. Zadaniem tej metody w klasach implementujących ten interfejs ma być wykonanie obliczeń na liczbach rzeczywistych i zwrócenie wyniku jako wartości typu `double`.

Następnie zdefiniuj publiczną abstrakcyjną klasę `Wyrazenie`, reprezentującą wyrażenie arytmetyczne pracujące na operandach (liczby i nazwane stałe), operatorach i funkcjach rzeczywistych. Klasa ta ma implementować interfejs `Obliczalny`; nie definiuj metody `oblicz()` w tej klasie, ponieważ jeszcze nie wiadomo co i jak należy policzyć – będzie to klasa bazowa dla innych klas realizujących konkretne obliczenia.

```
abstract class Wyrazenie {
    // ...
    /** metoda sumująca wyrażenia */
    public static int suma (Wyrazenie... wyr) { /* ... */ }
    /** metoda mnożąca wyrażenia */
    public static int iloczyn (Wyrazenie... wyr) { /* ... */ }
}
```

W klasie `Wyrazenie` umieść dwie statyczne metody ze zmienną liczbą argumentów, które będą realizowały zadanie sumowania i mnożenia wyrażeń.

Na koniec zdefiniuj klasy dziedziczące po klasie `Wyrazenie`, które będą reprezentowały operandy oraz różne funkcje i działania arytmetyczne. Operandy to: liczby i nazwane stałe. Operatory arytmetyczne to: dodawanie, odejmowanie, mnożenie, dzielenie oraz jedno-argumentowe operatory zmiany znaku na przeciwny ($x \mapsto -x$) i odwrotności ($x \mapsto 1/x$); popularne funkcje matematyczne to: sinus, cosinus, potęga, logarytm itp. Klasy operandów i operatorów powinny być tak zaprojektowane, aby można z nich było zbudować drzewo wyrażenia – obiekty klas `Liczba` i `Stała` to operandy, czyli liście w drzewie wyrażenia; natomiast operatory i funkcje to węzły wewnętrzne w takim drzewie – ich argumentami będą wyrażenia. We wszystkich klasach nadpisz metody `toString()` oraz `equals(Object)`.



Klasa `Liczba` ma przechowywać zwykłą wartość typu `double`; w klasie tej zdefiniuj dwa finalne i publiczne pola statyczne typu `Liczba` dla zera i jedności: `ZERO` (dla wartości 0.0) oraz `JEDEN` (dla wartości 1.0). Klasa `Stala` ma reprezentować nazwę skojarzoną z jakąś wartością; w klasie tej zdefiniuj dwa finalne i publiczne pola statyczne typu `Stala` dla popularnych liczb π i e, które są często używane w wyrażeniach arytmetycznych: `Pi` (π≈3.14) oraz `E` (e≈2.72).

Operatory jednoargumentowe to operatory prefiksowe, które będą zapisywane symbolicznie za pomocą tyldy ~ (zmiana znaku) i wykrzyknika ! (odwrotność). W operatorach dwu-argumentowych zaprojektuj system priorytetów, aby zminimalizować liczbę potrzebnych nawiasów przy wypisywaniu wyrażenia. Funkcje natomiast mają mieć swoje nazwy mnemoniczne.

Uzupełnij swoje zadanie o program testowy. W programie testowym skonstruuj drzewa obliczeń, wypisz je metodą `toString()` a potem wylicz ich wartości metodą `oblicz()` i wypisz otrzymane wartości. Przetestuj swój program dla następujących wyrażeń:

```
7 + 5 * x  -  1    # nazwa x oznacza stałą zdefiniowaną w programie
```

```
~ (2 - x) * e    # symbol ~ oznacza negację wyrażenia
```

```
(3 * π - 1) / (! x + 5)    # symbol ! oznacza odwrotność wyrażenia
```

```
sin((x + 13) * π / (1 - x))
```

```
exp(5) + x * log(e, x)
```

Na przykład wyrażenie `7 + x * 5` należy zdefiniować następująco:

```
Wyrazenie w = new Dodaj(
    new Liczba(7.2),
    new Mnoz(
        new Zmienna("x"),
        new Liczba(2.4)
    )
);
```

W programie testowym stałą `x` ustaw na wartość 1.618.