

KURS PROGRAMOWANIA W JAVIE

DRZEWA OBLICZEŃ

Instytut Informatyki Uniwersytetu Wrocławskiego

Paweł Rzechonek

Zadanie 1.

Zdefiniuj klasę **Para**, która będzie przechowywać pary klucz–wartość, gdzie klucz jest typu **String** a wartość typu **int**. Klucz powinien być polem publicznym ale niemodyfikowalnym a wartość polem ukrytym, które można odczytać za pomocą gettera i zmodyfikować tylko za pomocą settera.

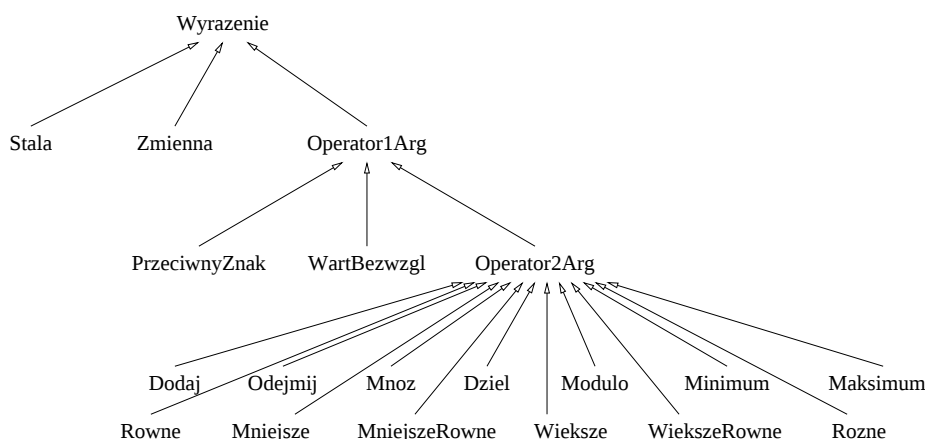
Zdefiniuj abstrakcyjną klasę **Słownik**, która będzie reprezentować zbiór obiektów typu **Para** z podstawowymi operacjami słownikowymi: wyszukiwanie (metoda **szukaj()**), wstawianie (metoda **wstaw()**), usuwanie (metoda **usuń()**) i policzenie wszystkich elementów w zbiorze (metoda **ile()**).

Następnie zdefiniuj klasę **Stos**, która będzie dziedziczyć po klasie **Słownik** i będzie realizować operacje słownikowe w oparciu o listę jednokierunkowej przechowując w węzłach wspomniane wcześniej pary (operacja **wstaw()** wstawia parę na początek listy, operacja **usuń()** usuwa parę z początku listy a operacja **szukaj()** przeszukuje całą listę). Klasa **Stos** powinna być tylko opakowaniem dla homogenicznej dynamicznej struktury danych opartej na węzłach. W klasie **Węzeł** zaimplementuj metody do wstawiania pary klucz–wartość na stos, usuwania pary i wyszukiwania w całym stosie wartości według zadanego klucza.

Zadanie 2.

Zdefiniuj abstrakcyjną klasę bazową **Wyrażenie**, reprezentującą wyrażenie arytmetyczne. W klasie tej umieść deklarację abstrakcyjnej metody **oblicz()**, której zadaniem w klasach potomnych będzie obliczanie wyrażenia i przekazywanie wyniku jako wartości typu **int**.

Następnie zdefiniuj klasy dziedziczące po klasie **Wyrażenie**, które będą reprezentowały kolejno liczbę (stała całkowitoliczbową), zmienną (zmienne pamiętają na zdefiniowanym wcześniej stosie par klucz–wartość), operacje arytmetyczne (dodawanie, odejmowanie, mnożenie, dzielenie i modulo oraz jednoargumentowa operacja zmiany znaku na przeciwny) i porównania (wynikiem porównania ma być jak w języku C liczba 0 albo 1 odpowiadająca wartościom logicznym *false* albo *true*). Klasy te powinny być tak zaprojektowane, aby można



z nich było zbudować drzewo wyrażenia: obiekty klas **Liczba** lub **Zmienna** to liście, a operatory to węzły wewnętrzne w takim drzewie. W klasach potomnych zdefiniuj metody **oblicz()** oraz **toString()**.

Na koniec napisz krótki program testowy, sprawdzający działanie obiektów tych klas. W swoim programie skonstruuj drzewa obliczeń, wypisz je metodą **toString()** a potem oblicz i wypisz wartość dla następujących wyrażeń:

```

3+5
2+x*7
(3*11-1)/(7+5)
((x+1)*x)/2
2*x+1<0

```

Na przykład wyrażenie $2+x*7$ należy zdefiniować następująco (klasa *St* przechowuje liczbę a klasa *Zm* nazwę zmiennej):

```

Wyrażenie w = new Wyrażenie(
    new Dodaj(
        new Liczba(2),
        new Mnóż(
            new Zmienna("x"),
            new Liczba(7)
        )
    )
);

```

Ustaw na początku programu testowego zmienną *x* na wartość -3 .

Zadanie 3.

Zdefiniuj abstrakcyjną klasę bazową *Instrukcja*, reprezentującą instrukcję w programie. W klasie tej umieść deklarację abstrakcyjnej metody *wykonaj()*, której zadaniem w klasach potomnych będzie wykonywanie odpowiednich obliczeń.

Następnie zdefiniuj klasy dziedziczące po klasie *Instrukcja*, które będą reprezentowały kolejno deklarację zmiennej (zmienne zapamiętują na stosie par klucz–wartość), instrukcję przypisania wartości wyrażenia do zmiennej, instrukcję warunkową (taką jak instrukcja *if/if-else* w języku C), instrukcję pętli (taką jak instrukcja *while/do-while* w języku C), przy czym warunki są wyrażeniami (warunek jest prawdziwy tylko wtedy gdy wartość wyrażenia jest różna od 0), instrukcje czytania ze standardowego wejścia i pisania na standardowe wyjście oraz instrukcję blokową (złożenie dwóch lub więcej instrukcji). Klasy te powinny być tak zaprojektowane, aby można z nich było zbudować drzewo sterujące sekwencją obliczeń. W klasach potomnych zdefiniuj metody *wykonaj()* oraz *toString()*.

Na koniec napisz krótki program testowy, sprawdzający działanie obiektów tych klas. W swoim programie skonstruuj drzewo obliczeń dla programu obliczającego silnię:

```

var silnia;
silnia = 1;
var n;
read n;
if (n>1)
{
    var i;
    i = 2;
    while (i<=n)
    {
        silnia = silnia*i;
        i = i+1;
    }
}
write silnia;

```

Wydrukuj swój program metodą *toString()* a potem wykonaj.

Uwaga.

W swoich programach nie czytaj ani nie analizuj danych ze standardowego wejścia. Drzewa obliczeń zdefiniuj na stałe w swoich programach testowych.