laboratorium: zadanie 4 *termin: 26/27 października 2011 r.*

KURS PROGRAMOWANIA W JAVIE

DRZEWA WYRAŻEŃ

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.

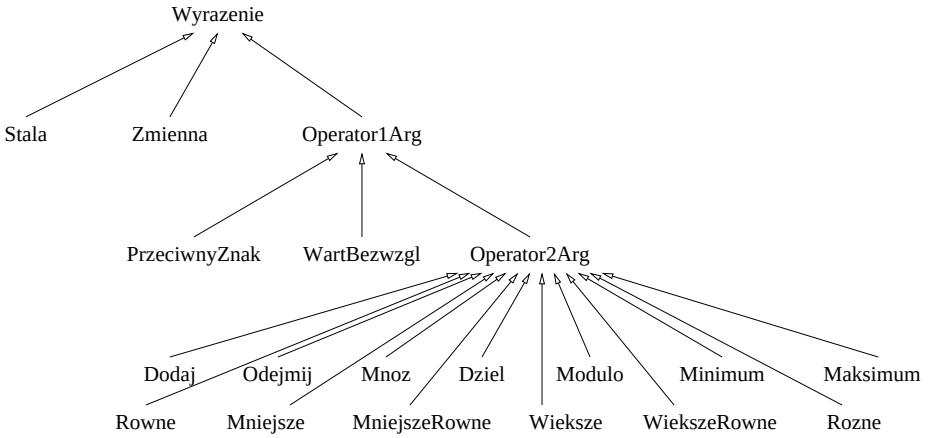
Zdefiniuj interfejs `Słownik`, który będzie reprezentowć dynamiczny zbiór obiektów typu `Para` z podstawowymi operacjami słownikowymi: wstawianie pary (metoda `wstaw()`), usuwanie pary o zadanym kluczu (metoda `usuń()`), wyszukiwanie wartości skojarzonej z zadanym kluczem (metoda `szukaj()`) i policzenie wszystkich elementów w zbiorze (metoda `ile()`).

Następnie zdefiniuj klasę `Zbiór`, która będzie implementować `Słownik` i będzie realizować operacje słownikowe w oparciu o listę, przechowując w węzłach wspomniane wcześniej pary. Możesz wykorzystać listę z poprzedniego zadania, modyfikując ją nieco i usuwając przy okazji zauważone wady. Klasa `Lista` może być komponentem składowym klasy `Zbiór`. Lista powinna być tylko opakowaniem dla homogenicznej dynamicznej struktury danych opartej na węzłach, zdefiniowanych jako wewnętrzna klasa `Węzeł`. W klasie `Węzeł` zaimplementuj metody do wstawiania pary klucz–wartość, usuwania i wyszukiwania pary względem zadanego klucza (można wykorzystać rekurencję).

Zadanie 2.

Zdefiniuj abstrakcyjną klasę bazową `Wyrazenie`, 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 `Wyrazenie`, które będą reprezentowały kolejno liczbę (stała całkowitoliczbowa), zmienną (wszystkie zmienne pamiętaj w polu statycznym typu `Zbiór` w zdefiniowanym wcześniej zbiorze par klucz–wartość), operacje arytmetyczne (dodawanie, odejmowanie, mnożenie, dzielenie i modulo oraz jednoargumentowa operacja zmiany znaku na przeciwny), porównania (wynikiem porównania ma być jak w języku C liczba 0 albo 1 odpowiadająca wartościom logicznym *false* albo *true*), itp. 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()`.

Zadanie 3.

Zdefiniowane w poprzednich zadaniach klasy umieść w pakiecie `narzędzia`. Na koniec napisz krótki program testowy (poza wymienionym pakietem), sprawdzający działanie obiektów klas wyrażenia. W swoim programie skonstruuj drzewa obliczeń, wypisz je metodą `toString()` a potem oblicz i wypisz ich wartość. Przeprowadź testy 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:

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

Ustaw na początku programu testowego zmienną `x` na wartość –3.

Uwaga 1.

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

Uwaga 2.

Program należy skompilować i uruchomić z wiersza poleceń!