Zadanie 3 *22–24 października 2014 r.*

KURS JĘZYKA JAVA

DRZEWA WYRAŻEŃ

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zadanie 1.

Zdefiniuj klasę `Para`, która będzie przechowywać pary klucz–wartość, gdzie klucz jest identyfikatorem typu `String` a skojarzona z nim wartość to liczba całkowita typu `double`. Klucz powinien być polem publicznym ale niemodyfikowalnym, a wartość polem ukrytym, które można odczytać za pomocą gettera i zmodyfikować tylko za pomocą settera.

```
public class Para {
    public final String klucz;
    private double wartość;
    // ...
}
```

W klasie tej zdefiniuj metody `toString()` oraz `equals(Object)` — dwie pary są równe gdy mają takie same klucze.

Zadanie 2.

Zdefiniuj klasę `Zbior`, która będzie przechowywać pary z zachowaniem warunku, że w zbiorze tym nie występują dwie pary o takim samym kluczu. Zaimplementuj zbiór na tablicy — długość tablicy ustaw na jakąś domyślną wartość (konstruktor bezparametrowy) lub pozwól prgramiście ustalić tą długość (kostruktor z odpowiednim parametrem). Operacje na zbiorze mają działać jak na stosie — dodajemy nowy element zawsze na końcu tablicy i usuwamy elementy tylko z końca. Dodatkowo należy zrealizować operację odnajdowania wartości na podstawie klucza.

```
public class Zbior {
    // ...
    /** metoda wstawia na koniec nową parę klucz-wartość */
    public void wstaw (String kl, double wart) throws IllegalArgumentException {
        /* ... */ }
    /** metoda odszukuje parę i zwraca wartość związaną z kluczem */
    public double czytaj (String kl) throws IllegalArgumentException {
        /* ... */ }
    /** metoda usuwa tyle pary z końca tablicy aby pzostało ich dokładnie s */
    public void usun (int s) throws IllegalArgumentException {
        /* ... */ }
```

```
    /** metoda usuwa jedną parę z końca tablicy */
    public void usun () throws IllegalArgumentException {
        /* ... */ }
    /** metoda podaje ile par jest przechowywanych w zbiorze */
    public int ile () {
        /* ... */ }
}
```

Implementując klasę `Zbiór` nie korzystaj z kolekcji standardowych.

Zadanie 3.

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 `double`.

Następnie zdefiniuj klasy dziedziczące po klasie `Wyrazenie`, które będą reprezentowały kolejno liczbę (stała zmiennopozycyjna), zmienną (wszystkie zmienne pamiętaj w polu statycznym typu `Zbior`), operacje arytmetyczne (dodawanie, odejmowanie, mnożenie i dzielenie oraz jednoargumentowe operacje zmiany znaku na przeciwny i odwrotności), popularne funkcje matematyczne, 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 tych podefiniuj metody `toString()` oraz `equals(Object)`.

Zadanie 4.

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 otrzymane wartości. Przetestuj swój program dla następujących wyrażeń:

```
3+5
2+x*7
(3*11-1)/(7+5)
arctan(((x+13)*x)/2)
pow(2,5)+x*log(2,y)
```

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

```
Wyrazenie w = new Wyrazenie(
    new Dodaj(
        new Liczba(7),
        new Mnoz(
            new Zmienna("x"),
            new Liczba(5)
        )
    )
);
```

Ustaw na początku programu testowego zmienną `x` na wartość `-1.618`.

Wskazówka

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

Uwaga.

Program należy skompilować i uruchomić z wiersza poleceń!