laboratorium: zadanie 5 *termin: 18 listopada 2009 r.*

KURS PROGRAMOWANIA W JAVIE

ABSTRAKCYJNE DRZEWA SKŁADNI DLA PROGRAMÓW

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zadanie 1.

Zdefiniuj publiczny interfejs `Obliczalny`, który będzie reprezentowć obiekty, które umieją policzyć wyrażenie całkowitoliczbowe metodą `oblicz()`. Metoda ta ma zwracać wartość typu `Integer`.

Zdefiniuj też publiczny interfejs `Wykonywalny`, który będzie reprezentowć obiekty, które umieją wykonać ciąg obliczeń metodą `wykonaj()`. Metoda ta nie powinna zwracać żadnej wartości.

Zaprojektuj i zdefiniuj własną hierarchię wyjątków potrzebną do realizacji poniższych zadań. Na szczycie tej hierarchii powinna się znaleźć publiczna klasa `Wyjatek` dziedzicząca po `Exception`.

Zadanie 2.

Zdefiniuj publiczną i abstrakcyjną klasę bazową `Wyrazenie` reprezentującą drzewo obliczeń dla całkowitoliczbowego wyrażenia arytmetycznego. Klasa ta powinna implementować intefejs `Obliczalny` bez definiowania metody `oblicz()`.

Następnie zdefiniuj całą hierarchię klas dziedziczących po klasie `Wyrazenie`, które będą reprezentowały: liczbę całkowitą, zmienną (zmienne mają być pamiętane w kolekcji `HashMap<String,Integer>` zadeklarowanej jako pole statyczne w klasie `Zmienna`), operacje arytmetyczne (dodawanie, odejmowanie, mnożenie, dzielenie, modulo i jednoargumentową zmianę znaku) oraz porównania (równe, różne, mniejsze, większe, mniejsze–równe i większe–równe). Obliczenie porównania ma zwracać wartość 1 albo 0, w zależności od tego czy warunek jest prawdziwy czy fałszywy. Przy obliczaniu dzielenia lub modulo trzeba zgłosić wyjątkek, gdy dzielnik ma wartość 0 (zdefiniuj własną klasę wyjątku). Konstruktory wymienionych klas powinny sprawdzać, czy ich argumenty są różne od `null`, a jeśli nie są to mają zgłosić odpowiedni wyjątek (zdefiniowany przez programistę). W klasach dziedziczących po `Wyrazenie` zdefiniuj metodę `oblicz()`.

Napisz krótki program testowy, sprawdzający działanie obiektów tych klas. Niech twój program obliczy i wypisze wartość następujących wyrażeń (ustaw zmienną x na wartość -1 a amienną y na wartość 2):

```
3+5
7*11+25/y
(3*11-1)/(7+5)
((x+1)*x)/2!=0
(2*x+1)%(x+1)<1==0
-(3*x*x+4*y*y)
z+x+y
```

Zadanie 3.

Zdefiniuj publiczną i abstrakcyjną klasę bazową `Instrukcja` reprezentującą drzewo obliczeń w programie. Klasa ta powinna implementować intefejs `Wykonywalny` bez definiowania metody `wykonaj()`.

Następnie zdefiniuj klasy dziedziczące po klasie `Instrukcja`, które będą reprezentowały: instrukcję blokową, deklarację zmiennej (zmienne zapamiętuj w kolekcji `HashMap<String,Double>` i inicjalizuj wartością 0), instrukcję przypisania wartości wyrażenia do zmiennej, instrukcje warunkowe (takie jak instrukcje `if` oraz `if-else` w języku C), instrukcje pętli (takie jak instrukcje `while` i `do-while` w języku C), instrukcję czytania ze standardowego wejścia oraz instrukcję pisania na standardowe wyjście. Warunek w instrukcji warunkowej lub w pętli jest wyrażeniem (warunek jest prawdziwy tylko wtedy gdy wartość wyrażenia jest różna od 0). Instrukcja blokowa powinna być inicjalizowana dowolną liczbą instrukcji wewnętrznych (konstruktor ze zmienną liczbą argumentów); pozatym instrukcja ta ma spamiętywać jakie zmienne zostały utworzone w tym bloku i na końcu ma je usunąć (nie wolno tworzyć zmiennych o takich samych nazwach). Konstruktory wymienionych

klas powinny sprawdzać, czy ich argumenty są różne od `null`, a jeśli nie są to mają zgłosić odpowiedni wyjątek. W klasach dziedziczących po `Instrukcja` zdefiniuj metodę `wykonaj()`.

Napisz krótki program testowy, sprawdzający działanie obiektów tych klas. Niech twój program sprawdzi, czy wczytana liczba jest pierwsza. Program ten może działać według następującego schematu:

```
var n;
read n;
if (n<2) write 0;
else
{
   var p;
   var wyn;
   p = 2;
   while (p*p<=n)
   {
      if (n%p==0)
      {
         wyn = p;
         p = n;
      }
      p = p+1;
   }
   if (wyn>0) write 0;
   else write 1;
}
```

Uwaga 1.
Wszystkie wyjątki, interfejsy i klasy (oprócz klas z programami testowymi) umieść w pakiecie `obliczenia`.

Uwaga 2.
W każdej klasie z pakietu `obliczenia` zdefiniuj metodę `toString()`, która będzie potrafiła zrekonstruować tekstową postać wyrażenia czy programu.

Uwaga 3.
Umieść w swoim programie co najmniej jedną asercję.