Projekt 1: analiza leksykalna MinimL

1. Wstęp.

Celem niniejszego projektu jest napisanie analizatora leksykalnego prostego języka MinimL.
Ogólne informacje o języku MinimL i jego strukturze leksykalnej znajdziecie tutaj:
`http://www.cs.geneseo.edu/~baldwin/csci331/spring2003/miniml.html`
Z kolei dokumentacja dotycząca klas potrzebnych przy pisaniu kompilatora jest do znalezienia tutaj:
`http://www.cs.geneseo.edu/~baldwin/miniml/doc/index.html`

2. Zadanie

Należy napisać analizator leksykalny dla MinimL.

W tym semestrze z westchnieniem ulgi zapominamy o *lex*'ie, *flex*'ie i im podobnych i we wszystkim, co będziemy robić, będziemy poruszać się w modułach Sun Java. Dzisiejszy projekt będzie to pierwszy moduł w projekcie, w którym w końcowym efekcie napiszemy kompilator języka MinimL. W odniesieniu do klas opisanych na stronie Baldwina, Wasze główne zadanie będzie polegało na stworzeniu implementacji klasy MinimLTokenSeq. Jeżeli Wasz analizator leksykalny będzie miał dokładnie taki sam interfejs jak opisywana klasa (łącznie z jej nazwą), to powienien być całkowicie kompatybilny z dowolnym kodem w języku MinimL użytym potem do testowania.

Prawdopodobnie będziecie też chcieli napisać klasę analogiczną do MinimLTransducer. Nic z tej klasy nie powinno być widoczne dla reszty Waszego kompilatora, nie trzeba zatem "kopiować" toczka w toczkę klasy MinimLTransducer tak, jak w przypadku klasy MinimLTokenSeq, ale będziecie potrzebowali czegoś, co będzie działało podobnie, jak ta klasa. Lektura dokumentacji też może okazać się pomocna.

Generalnie wygodnie będzie zacząć zadanie od podejrzenia klas Baldwina i zrozumienia, która z nich co robi.

Last but not least, klasy Baldwina są do ściągnięcia z jego strony:
`http://www.cs.geneseo.edu/~baldwin/miniml/`
oraz, na wszelki wypadek, także stąd:
`http://www.math.us.edu.pl/~pgladki/teaching/2011-2012/tk2-miniml.tar`