Aihe: Lisp-tulkki

Suunnitellaan ja toteutetaan toimiva, yksinkertainen Lisp-tulkki Java-kielellä. Tulkki sisältää Lisp-kieliperheeseen kuuluvan Scheme-kielen olennaisimmat osat: read-eval-print -silmukan sekä ainakin suurimman osan, jos ei kaikkia, R6RS:n mukaisista primitiivifunktioista/ proseduureista. Funktioilla on kaikkien Lisp-varianttien mukaisesti first-class -ominaisuus siten, että niitä voidaan antaa argumentteina toisille funktioille. Tulkki sisältää tuen myös anonyymien funktioiden luontiin ja käyttöön lambda-käskyn avulla. Tulkki ei sisällä call-with-current-continuationeja eikä makroja. Tulkki sisältää ennaltamääritellyn Globalenvironmentin, sekä mahdollisuuden laajentaa sen pohjalta paikallisia ympäristöjä funktioiden määrittelyjä sekä käyttöä varten.

Käyttäjät: Lisp-ohjelmoija
Käyttäjien toiminnot
Syöttää tulkille SkeletonLisp-lauseita (L-expressions), joiden arvoja tulkki evaluoi.

hyväksyttäviä Lisp-lauseita ovat:
* nil: (“nil")
* merkkijono: “tämä merkkijono sisältää kirjaimia ja alku-ja loppusitaatit"
* atomit : (‘a, ‘atomi, ‘terve-76)
* kokonaisluvut: (56, -254 : rajoituksena samat kuin Javan int-tyypin rajoitukset)
* liukuluvut: (323.546, -0.3567: rajoituksena samat kuin Javan double-tyypin rajoitteet)
* ID-t: (f, add, sqrt, user-defined-id), joitakin rajoituksia sisältyy Id:itten käyttöön, kuten se, että ne eivät see alkaa numeromerkillä.
* lambda-lauseet, jotka voivat olla joko muotoa:
  (lambda (var-1 ... var-n) body) tai:
  (lambda list-param body)
* “applikaatiot” (applications), jotka voivat olla jotakin seuraavista:
  1. (Id val-1 ... val-n)
  2. (lambda-expression val-1 ... val-n)
  3. (application val-1 ... val-n)
* cond-lause:

```
(cond   (ehto-1 arvo-1)
        ...
        (ehto-n arvo-n))
```

[cond-lauseen ei tarvitse sisältää else-osiota, mutta mikäli tätä ei ole, eikä mikään ehdoista toteudu, ilmoittaa tulkki virhetilanteesta käyttäjälle, eikä lausetta pystytä evaluoimaan.

“list”-applikaation avulla luodaan uusia LPair-tyyppejä, jotka sisältävät kaksi alkiota: ((car LExp) (cdr LExp)), ja LPair:ien tulostusmuoto riippuu siitä, minkälainen lause (cdr LExp) on; jos se on jotain muuta kuin toinen pari, tai nil, tulostusmuoto on:
((car LExp) **.** (cdr LExp)),
jos se on toinen LPair, tulostusmuoto on:
((car LExp) (cdr LExp)),
muuten (nil:llä) sen muoto on:
((car LExp))

“error”-applikaatiolla määritellään LError-tyyppejä, joiden arvo on:
“<error>: message”