hyväksymispäivä arvosana

arvostelija

COBOL ja Python: Datan kapselointi

Erkki Heino, Tero Huomo, Eeva Terkki

Helsinki 19.2.2013

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

1 Datan kapselointi **1**

1.1 COBOL 1

1.1.1 Object-Oriented COBOL 2

1.2 Python 3

2 Etuja ja haittoja **3**

Lähteet **4**

1 Datan kapselointi

1.1 COBOL

COBOLissa tietoa voidaan ryhmitellä muodostamalla tietueita (record) ja ryhmiä (group). `DATA DIVISION` -osiossa käytettävät muuttujien tasonumerot (level number) määrittävät, minkälaisia kokonaisuuksia muuttujat muodostavat. Seuraavassa esimerkissä on kuvattu opiskelijatietue.

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 OPISKELIJA.
   05 NIMI.
       10 ETUNIMI  PIC A(10).
       10 SUKUNIMI PIC A(10).
   05 NUMERO       PIC 9(10).
   05 SUKUPUOLI    PIC A.
       88 MIES     VALUE "M".
       88 NAINEN   VALUE "N".
```

`OPISKELIJA` on tietue, joka koostuu nimestä, opiskelijanumerosta ja sukupuolesta. Nimi on jaettu kahteen osaan: etunimi ja sukunimi.

Tasonumerot kuvaavat tiedon hierarkian, mutta tietyillä numeroilla on erityismerkitys. Hierarkiaa kuvaavat tasonumerot 02-49. Tasonumero 01 tarkoittaa, että kyseessä on tietue. Merkitsemällä tasonumeroksi 66 voidaan edeltävälle muuttujalle antaa toinen nimi. 77 kuvaa muuttujaa, joka ei ole osa mitään rakennetta. Tason 88 avulla voidaan määrittää muuttujaan liittyviä ehtoja: yllä olevassa esimerkissä opiskelijan sukupuoli on mies, jos muuttujan `SUKUPUOLI` arvo on M, ja nainen, jos arvo on N [Ans74, s. I-84]. Esimerkiksi voitaisiin tarkistaa, onko opiskelija mies:

```
IF MIES OF OPISKELIJA
```

Todellisuudessa tietue ei ole juuri muuta kuin merkkijono, jonka eri osat kuvaavat tietueen kenttiä. Tietueelle voidan antaa arvot kenttä kerrallaan:

```
MOVE "ARTO" TO ETUNIMI OF OPISKELIJA.
```

Toisaalta koko tietueelle voidaan antaa arvo kerralla:

```
MOVE "ARTO      WIKLA     1234567890M" TO OPISKELIJA.
```

Esimerkin tietue sisältää kerrallaan yhden opiskelijan tiedon — tällaisenaan koodissa ei siis voida käsitellä useampaa opiskelijaa samanaikaisesti. Rakenne voidaan määritellä taulukoksi (table), jolloin samaa rakennetta voidaan käyttää useamman samanrakenteisen tiedon käsittelemiseen ilman saman rakenteen uudelleenmäärittelyä.

Muuttuja määritellään taulukoksi `OCCURS`-lauseen avulla. `OPISKELIJA` voitaisiin muuttaa taulukoksi muokkaamalla määrittelyä seuraavasti:

```
01 OPISKELIJA OCCURS 5 TIMES.
```

Esimerkiksi taulukon toiseen opiskelijaan viitattaisiin indeksillä 2: `OPISKELIJA(2)`. Taulukon sisällä voidaan myös määritellä sisäkkäisiä taulukoita.

1.1.1 Object-Oriented COBOL

COBOL ei alunperin ole olio-ohjelmointikieli. COBOLista on kuitenkin kehitetty olio-ohjelmoinnin mahdollistava laajennus, Object-oriented COBOL. Ensimmäiset oliolaajennukset COBOLiin tehtiin jo vuonna 1997 [Wik13]. Kieltä laajensivat esimerkiksi Micro Focus sekä IBM. Lopullinen ISO standardi Object-Oriented COBOLille tehtiin vuonna 2002.

Object-Oriented COBOL (jatkossa OO COBOL) lisää kieleen kolme olio-ohjelmointikielen perustavanlaatuista ominaisuutta - tiedon kapseloinnin, perinnän sekä polymorfismin. Olio-ohjelmoinnin lisäyksien syntaksi sekä tarjotut ominaisuudet vaihtelevat suuresti eri toteuttajien välillä. Niiden yhtenäisyys vuoden 2002 ISO standardin kanssa on vaihteleva.

Olio-laajennoksen tärkeimpiä lisäyksiä on mahdollisuus luokkien ja olioiden määrittelyyn. Kuten normaali COBOL-ohjelma, luokka määritellään neljän *osion* (division) avulla. IBM'n toteutus [IBM13] ja Micro Focus'n toteutus [Mic13] ovat hyvin samankaltaisia. `IDENTIFICATION DIVISION`-osiossa ohjelman tunnuksen sijasta annetaan luokan tunnus, `CLASS ID`, sekä mahdolliset periytymisestä kertovat tiedot. Luokan yhteinen tietosisältö ja metodit luetellaan `FACTORY`-osiossa, olioiden tietosisältö ja metodit `OBJECT`-osiossa. `FACTORY` ja `OBJECT` -osiot jakautuvat edelleen vanhemmista COBOL-versioista tuttuihin `DATA DIVISION` ja `PROCEDURE DIVISON` osioihin.

Olioiden metodien kutsumiseksi on OO COBOLiin lisätty avainsana `INVOKE`.

```
INVOKE ARTONTILI "TALLETA" USING KATEINEN RETURNING KUITTI
```

Esimerkissä kutstutaan olion `ARTONTILI` metodia `TALLETA`, jolle annetaan parametriksi muuttujan `KATEINEN` arvo. Metodin palauttama arvo sijoitetaan muuttujaan `KUITTI`.

Periminen ja rajapintojen toteutus vaihtelee runsaasti. Micro Focus'n Visual COBOL ei mahdollista moniperintää, mutta luokat voivat toteuttaa Javan tyyliin usean rajapinnan [Mic13]. IBM:n Enterprise COBOL ei määrittele dokumentoinnissa rajapintoja lainkaan, mutta estää moniperinnän [IBM13]. Michael Kastenin [Kas98] mukaan C++:n tavoin IBM:n OO COBOL kuitenkin luettelisi erilaisia sääntöjä, joilla usean luokan yhtäaikaista perimisestä kumpuava niin kutsuttu timanttiongelma ratkaistaan. Tästä voidaan päätellä, että OO COBOL on myös tukenut moniperintää. Myös Micro Focus'n varhainen dokumentaatio mainitsee moniperinnän tuen [Mic06].

1.2 Python

2 Etuja ja haittoja

Lähteet

Ans74 *American national standard programming language COBOL.* ANSI, New York, USA, 1974.

IBM13 Writing object-oriented programs, `http://pic.dhe.ibm.com/infocenter/pdthelp/v1r1/index.jsp?topic=%2Fcom.ibm.entcobol.doc_4.1%2FPGandLR%2Ftasks%2Ftpoot02.htm`. [19.2.2013].

Kas98 Oo cobol: Comparison to c++, howpublished =.

Mic13 Micro focus cobol language reference, `http://documentation.microfocus.com/help/index.jsp?topic=%2Fcom.microfocus.eclipse.infocenter.visualcobol.vs%2FHRLHLHPDFX03.html`. [19.2.2013].

Mic06 Oo programming with object cobol, `https://supportline.microfocus.com/Documentation/books/nx50/opconc.htm`. [19.2.2013].

Wik13 Cobol, `http://en.wikipedia.org/wiki/COBOL`. [19.2.2013].