hyväksymispäivä arvosana

arvostelija

Olio-ohjelmointikielten historia

Matias Käkelä

Helsinki 14.2.2006

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Matemaattisluonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author

Matias Käkelä

Työn nimi — Arbetets titel — Title

Olio-ohjelmointikielten historia

Oppiaine — Läroämne — Subject

Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Seminaaritutkielma	14.2.2006	16 sivua + 1 liitesivua

Tiivistelmä — Referat — Abstract

Olio-ohjelmointi on yksi käytetyimmistä ohjelmointiparadigmoista ja sen vuoksi nykyään suosituimmat ohjelmointkielet ovat oliopohjaisia. Vaikka olioperustainen ohjelmointi on tullut yleiseen tietoisuuteen vasta viime vuosikymmenellä, sen historia ylettyy yli neljänkymmenen vuoden taakse. Tässä seminaaritutkielmassa tarkastellaan olio-ohjelmointikielten kehitystä niiden kielten kautta, jotka olivat keskeisesti vaikuttamassa olioajattelun kehitykseen ja laajamittaiseen käyttöönottoon. Tutkielmassa esitellään ensimmäinen oliokieli Simula, olioajattelua kehittänyt Smalltalk, olio-ohjelmoinnin ensimmäinen todellinen menestyjä C++ sekä nykyään johtoasemassa oleva oliokieli Java.

ACM Computing Classification System (CCS):
K.2 [History of Computing: Software],
D.3.2 [Language Classifications: Object-oriented languages]

Avainsanat — Nyckelord — Keywords

olio-ohjelmointi, oliokielten historia, ohjelmointikielten historia

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Yksinkertaisimmillaan tietokoneohjelma voidaan pelkistää peräkkäisiksi toisiaan seuraaviksi komennoiksi. Kun komennot yhdistetään yhdeksi kokonaisuudeksi, muodostuu ohjelmaksi kutsuttu toiminnallinen yksikkö. On kuitenkin useita ajattelutavaltaan erilaisia tapoja luoda komennot yhdistävä kokonaisuus. Tässä seminaaritutkielmassa esitellään olioperustainen lähestymistapa ohjelmointiongelmien ratkaisemiseen.

Kautta ohjelmointikielten historian on pyrkimyksenä ollut kielten abstraktiotason nostaminen. Abstraktiotason korottamisella pyritään ohjelmoinnista tekemään ymmärrettävämpää ja helpommin hallittavaa. Ennen olio-ohjelmointikielten syntyä käytetyillä perinteisillä ohjelmointimenetelmillä oli vaikeaa ja kallista hallita laajoja kokonaisuuksia. Perinteisiä ohjelmointikieliä pidetään myös oliokieliä vaikeampina opetella [Soe05].

Olioparadigman kehittyminen 1960-luvulla toi täysin uuden näkökulman ohjelmointiin. Perinteisestä rakenteisesta ohjelmoinnista siirryttiin olioajatteluun, jossa ohjelman toiminnallisuus jaetaan olioihin. Yksittäiset oliot muodostavat yhdessä toimiessaan varsinaisen ohjelman [Ren82]. Rakenne on osoittautunut niin hyväksi, että suosituimmat nykyaikaiset ohjelmointikielet ovat oliokieliä. Perinteisiä ohjelmontikieliä käytetään yhä paljon, mutta oliopohjaiset kielet ovat saavuttaneet johtoaseman [Wil96].

Tutkielmassa tutustutaan olio-ohjelmointikieliin esittelemällä keskeisimmät kielet, joita oliokielten kehityksen taipaleella on syntyntyt. Lukijalle esitellään tärkeimmät olio-ohjelmointikielet sen historian murrosvaiheilta aina ensimmäisestä oliokielestä Simulasta ja tämän jälkeläisestä Smalltalkista nykyään suosittuihin Java- ja C++ -kieliin.

2 Ennen oliopohjaisia kieliä

Luku esittelee syyt, jotka johtivat korkean tason kielten kehittymiseen. Luvussa tutustutaan myös olio-ohjelmointikieliä edeltäneisiin ohjelmointikieliin, jotka vaikuttivat oliokielten syntyyn.

2.1 Kohti korkeamman tason kieliä

Tietokoneiden historian alkuaikoina tietokoneita ohjelmoitiin konekielellä, joka oli hyvin virhealtista ja vaikeaa. Kun konekieliset ohjelmat kasvoivat, niiden ylläpito vaikeutui huomattavasti. Sen vuoksi tarvittiin korkean tason kieliä, joiden ohjelmointi olisi konekieliä yksinkertaisempaa.

Ensimmäisiä korkean tason kieliä olivat mm. Algol ja Fortran, jotka perustuivat rakenteiseen ohjelmointiin. Rakenteiset kielet pyrkivät osittamaan ongelmat pienempiin osaongelmiin ja ne mahdollistivat aiempaa joustavamman ohjelmoinnin [Wik98, s. 5–8]. Konekielen käskyt vaihtuivat ymmärrettävämmiksi ja lyhyemmiksi lausekkeiksi. Samalla muuttujat korvasivat muistiosoitteet ja monimutkaisten tietorakenteiden käsittely helpottui [Cap03].

Konekieleen verrattuna uudet kielet toivat myös muita uusia ominaisuuksia ohjelmakoodin hallitsemiseen. Ne mahdollistivat algoritmien jakamisen erillisiin alialgoritmeihin. Algoritmien jakaminen pienempiin osakokonaisuuksiin helpotti ohjelmointiongelman jäsentelyä, niin että oikean toimintalogiikan ohjelmointi oli helpompaa ja tuotettu ohjelmakoodi oli mahdollisimman laadukasta. Osiin jakaminen helpotti myös uudelleenkäyttöä. Aliohjelmia ja moduuleita voitiin tarvittaessa käyttää suoraan tai vähäisin muutoksin toisten ohjelmien osina.

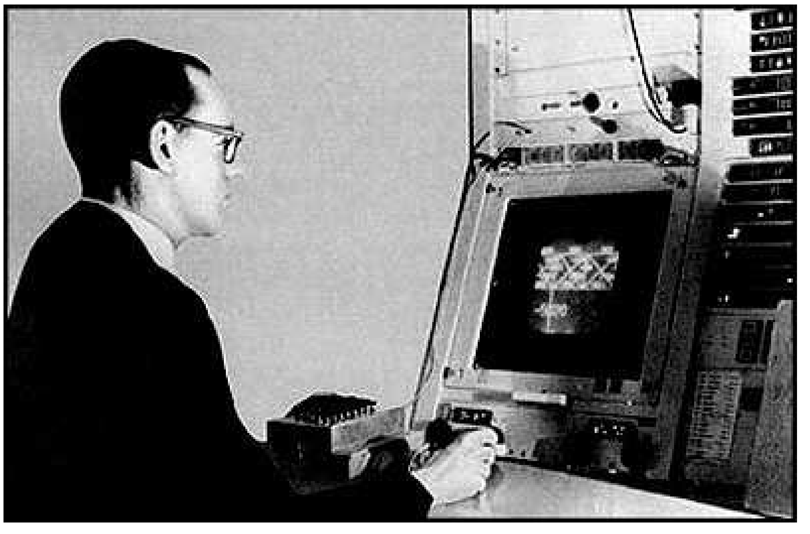
Vaikka siirtyminen konekielestä korkean tason kieliin muutti ja helpotti ohjelmointia huomattavasti, pian havaittiin, että rakenteiselle ohjelmoinnille oli olemassa varteenotettavia vaihtoehtoja.

2.2 Murrosvaihe

Ensimmäiset vihjeet uudenlaisesta lähestymistavasta ohjelmointiin tulivat 1960-luvun alussa. Eräs olioajattelun pioneereistä oli Ivan Sutherland, joka kehitti 1963 Sketchpad-ohjelman. Vaikka Sketchpad ei ollut ohjelmointkieli, sen merkitys oli suuri. Se oli ensimmäinen tietokoneohjelma, joka esitteli niin graafisen käyttöliittymän kuin olioihin perustuvan ajattelutavankin [Kay93]. Sketchpad näytti ensimmäisenä, miten asioita voidaan käsitellä olioina. Se loi ajatusmallin, jossa oliot olivat abstrakteja tietorakenteita. Ne kapseloivat sisältämänsä tiedon, ja näitä tietoja manipuloitiin määriteltyjen rajapintojen kautta.

Sketchpadia voidaan siten pitää yhtenä olioparadigman ensimmäisistä kehittäjistä. Sen suuresta merkityksesta on kertonut esimerkiksi Smalltalk-kielen kehittäjä Alan

C. Kay, jonka mukaan Sutherland oli keksinyt täysin uudenlaisen ja innovatiivisen tavan hyödyntää tietokonetta [Kay93].



Kuva 1: Ivan Sutherland ja Sketchpad
[http://features.cgsociety.org/stories/2003_10/rendering/sutherland.jpg]

3 Olio-ohjelmointikielten synty

Kun Sketchpad oli näyttänyt suuntaa uudenlaisen ajattelutavan rakentumiselle, alkoi ensimmäisten oliokielten kehitys. Oliopohjaisessa ohjelmointitavassa nähtiin useita etuja perinteiseen ohjelmointiin verrattuna.

Olio-ohjelmoinnin perusajatus on maailman ilmiöiden kuvaaminen olioina. Oliot kapseloivat sisäänsä tiedon omasta tilastaan ja toiminnastaan, ja niitä muokataan useinmiten vain ulkoa käsin. Kuten rakenteisessa ohjelmoinnissa, oliokielissäkin ongelma-alue voidaan jakaa useisiin pieniin osaongelmiin. Olio-ohjelmoinnissa erotetaan rakenne ja toiminnallisuus toisistaan selkeämmin kuin rakenteisessa ohjelmoinnissa. Jokainen olio on ideaalitilanteessa erillinen itsenäinen yksikkö, joka joko tarjoaa tai tuottaa palveluita muille olioille. Kun yksittäiset oliot liitetään yhteen, ne muodostavat keskenään rakenteen, joka toteuttaa halutun toiminnallisuuden [Wik98, s. 5 – 8].

Olioajattelun kehittyminen oli merkittävä askel ohjelmointikielten historiassa. Olioiden parempi vastaavuus ympäröivään maailmaamme on ollut myös yksi keskeisistä syistä oliokielten suosioon. Oliotekniikoilla pystytään maailman todelliset ilmiöt kuvaamaan entisiä tekniikoita joustavammin sekä ymmärrettävämmin. Yksi tärkeimmistä olio-ohjelmoinnin alkuperäisistä tavoitteista oli helpottaa todellisen maailman ongelmien ratkaisemista luomalla perinteisiä ohjelmointikieliä soveltuvammat välineet niiden mallintamiseen [Cap03]. Olio-ohjelmointi toi myös muita etuja. Oliopohjaisuus mahdollisti mm. ohjelman joustavamman jakamisen osiin sekä olioiden tai oliokokonaisuuksien edistyneemmän hyödyntämisen uudelleenkäytössä [Cap03].

3.1 Ensimmäinen oliokieli: Simula

Simula I ja Simula 67 olivat ensimmäiset olio-ohjelmointikielet. Simula-kielen kehittivät norjalaiset Ole-Johan Dahl ja Kristen Nygaard Norjan tieteellisessä laskentakeskuksessa Oslossa. Vaikka Simula I oli vielä osin puutteellinen olio-ohjelmointikielenä, sen seuraaja, Simula 67, sisälsi suurimman osan nykyaikaisten oliokielten kalustosta. Sen ominaisuuksiin kuuluivat niin luokat ja oliot kuin periytyminenkin [DaN05].

Simula-kieltä alettiin alunperin kehittää reaalimaailman ongelmien simulointia varten. Kielen juuret ovat Algol-kielessä, jonka pohjalle Simula I-kieli kehitettiin [Kay93]. Simula I-kieltä pidettiin lähinnä simulaatioiden laatimiseen soveltuvana ohjelmointikielenä, eikä sen oliopohjaisuus ollut vielä kehittynyt. Simula 67:ään tuli olioajattelun kannalta tärkeitä lisäominaisuuksia. Esimerkiksi perintä tuli kieleen vasta Simula 67:ssä, jonka myötä kieli laajeni simulaatioista yleiskäyttöiseksi ohjelmointikieleksi [DaN05].

Ensimmäisiä Simula-kääntäjiä eri alustoille alkoi ilmestyä 1970-luvun alussa. Simula-kääntäjiä tehtiin mm. UNIVAC:lle, IBM:lle ja Burroughsille. Vaikka Simula 67-kieltä on käytetty ja käytetään yhä edelleen todellisiinkin tarpeisiin, se on merkittävä kieli erityisesti sen vuoksi, että se oli ensimmäinen olioperustainen ohjelmointikieli. Kuvassa 2 esitellään Simula-kiellellä toteutettu ohjelma.

3.2 Olioajattelun kehittäjä: Smalltalk

Vaikka Simula olikin ensimmäinen olio-ohjelmointikieli, Smalltalkia pidetään usein kielenä, joka valjasti olio-ohjelmoinnin täydellisesti käyttöönsä, ja jalosti sen nykyiselleen. Se oli ensimmäinen ohjelmointikieli, joka toteutti kaikki modernien oliokiel-

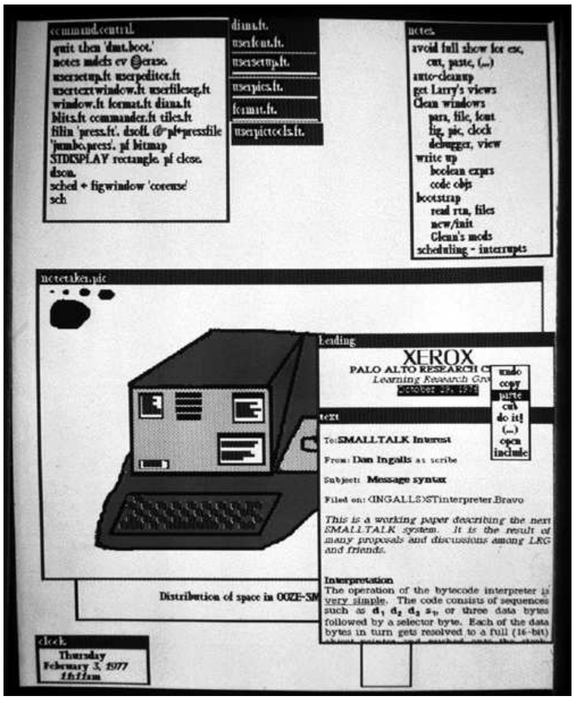
```
Begin
 Class Greetings;
 Begin
  OutText("Hello World!");
  OutImage;
 End;

 REF(Greetings) hello;
 hello :- New Greetings;

End of program;
```

Kuva 2: Luokkarakennetta hyödyntävä Hello World-ohjelma Simula 67-kielellä toteutettuna

ten piirteet. Smalltalkin merkityksestä kertoo myös se, että olioperustainen ohjelmointi sai nimensä kieltä kehitettäessä: termiä *object oriented* käytettiin ensimmäistä kertaa Smalltalkin kehityksen aikana [Ren82]. Smalltalk on edelleen merkittävä ohjelmointikieli, sillä sen oliomalli on puhtaampi ja yhtenäisempi kuin useimmissa nykyisissä oliokielissä. Smalltalkia voidaan myös pitää nykyisten ikkunoihin perustuvien graafisten käyttöliittymien isänä, sillä se ei ollut pelkästään ohjelmointikieli, vaan myös graafinen ohjelmointiympäristö.

Smalltalk-kieltä alettiin kehittää 1960 ja 1970 -lukujen taitteessa Xeroxin Palo Alton tutkimuskeskuksessa. Alan C. Kay työskenteli keskuksessa Dynabook-projektissa, jonka tarkoituksena oli kehittää monikäyttöinen, edullinen ja helppokäyttöinen pientietokone. Ennen Xeroxia Kay oli työskennellyt Utahin yliopistossa, jossa hän oli tutustunut Sketchpad-laitteeseen ja Simula-kieleen. Näiden vaikutus näkyy Smalltalk-kielessä, vaikka se lainasi myös LISP-kielestä. Simulan oliomalli ei kuitenkaan ollut riittävä Smalltalkille, vaan Kay kehitti sitä pidemmälle. Toisin kuin Simulassa, Smalltalkissa kaikki asiat ovat olioita. Smalltalkin oliot kommunikoivat keskenään lähettämällä ja vastaanottamalla viestejä ja aivan kuten esim. Javassakin, jokainen Smalltalkin luokka perii yhteisen Object-luokan [Tun98]. Kuvassa 4 esitellään Smalltalk-kielen syntaksia.

Smalltalk-kielen ensimmäinen kokeellinen versio valmistui lokakuussa 1972. Se oli



Kuva 3: Smalltalk-72-ohjelmointiympäristön käyttöliittymä [http://squab.no-ip.com:8080/collab/uploads/61/smalltalk-72-1977.jpg]

muutaman tuhannen rivin pituinen Basic-kielinen ohjelma, joka osasi laskea laskutoimituksen 3 + 4 summan. Smalltalkin ensimmäinen todellinen versio ilmestyi kaksi kuukautta myöhemmin, ja se sai nimekseen Smalltalk-72. Seuraava kielen versio oli Smalltalk-74, jonka jälkeen ilmestynyt Smalltalk-76 laajensi kielen olio-ominaisuuksia. Se toi mukanaan mm. periytymisen. Samalla kieli siirtyi siirrettävyyden vuoksi käyttämään tavukoodipohjaista arkkitehtuuria ja sille kehitettiin emulaattori, joka suoritti tavukoodia. Smalltalkista ilmestyi vielä Smalltalk-78-versio, jonka jälkeen ilmestynyt Smalltalk-80 julkaistiin ensimmäistä kertaa Xeroxin ulkopuolella. Smalltalk oli tuonut olio-ohjelmoinnin ensimmäistä kertaa yleiseen tietoisuuteen [Tun98].

```
VisualComponent subclass: #HelloWorldView
instanceVariableNames: ''
classVariableNames: ''
poolDictionaries: ''
category: 'test'

displayOn: aGraphicsContext

'Hello World!' asComposedText displayOn: aGraphicsContext.

open

|window|
window := ScheduledWindow new.
window label: 'Hello World Demo:'.
window component: self new.
window open.
```

Kuva 4: Hello World-ohjelma Smalltalk-kielellä esitettynä. Ohjelma esittää tekstiä graafisessa käyttöliittymäikkunassa.

4 Oliokielten esiinmarssi

Luku käsittelee olio-ohjelmoinnin nousua hallitsevaksi ohjelmointimenetelmäksi. Luvussa esitellään suosituimmat oliopohjaiset ohjelmoinitkielet ja kerrotaan syistä, joilla ne ovat saavuttaneet nykyisen asemansa.

4.1 Ensimmäinen suurmenestys: C++

Smalltalkin myötä olioperustaisen ohjelmoinnin suosio oli alkanut kasvaa. Oliokielet eivät olleet kuitenkaan vielä saavuttaneet laajaa kaupallista menestystä. Eräänä syynä heikkoon menestykseen voidaan pitää ohjelmistokehittäjien haluttomuutta vaihtaa radikaalisti työvälineitään. Ajattelutavaltaan täysin uuteen ohjelmointikieleen siirtyminen on kallista ja aikaavievää: vanhaa ohjelmakoodia ei voida enää hyödyntää ja kieli pitää opetella täysin alusta aina perusrakenteitaan myöten [Sut99]. C++ oli ratkaisu tähän ongelmaan. Se lisäsi olio-ominaisuudet C-kieleen, joka oli hyvin suosittu ja pitkään käytössä ollut ohjelmointkieli. C++ piti uusista ominaisuuksista huolimatta aiemmin laaditun ohjelmakoodin muutamia poikkeuksia lukuunottamatta täysin yhteensopivana uuden kanssa.

Bjarne Stroustrup alkoi kehittää C++ -kieltä 1979 Bellin laboratoriossa AT&T-yhtiössä. Hän oli todennut Simulan esittelemistä oliotekniikoista olevan apua laajoissa ohjelmistoprojekteissa, mutta toisaalta sen olevan liian hidas käytännön tarpeisiin. C-kielen edeltäjä BCPL-kieli tuotti tehokkaita ohjelmia, mutta Stroustrup piti sitä liian laiteläheisenä. Pettyneenä olemassa oleviin ohjelmointivälineisiin Stroustrup alkoi kehittää C++ -kieltä C-kielen pohjalle [Str93]. C-kieli oli eräs 1980-luvun vaihteen käytetyimpiä ohjelmointikieliä. C-kielen suosiota selittivät monet seikat: C-kieliset sovellukset olivat nopeita, sillä ne käännettiin suoraan konekielelle. C-kieli oli myös joustavaa sekä helposti siirrettävää. C-kielen suosion keskeisimpänä syynä oli kuitenkin sen symbioosi Unix-käyttöjärjestelmän kanssa [Sut99].

Alussa Stroustrup kehittämää kieltä kutsuttiin nimellä C with Classes. Stroustrup laati sille Cfront-nimisen kääntäjän, joka todellisuudessa vain käänsi C++ -koodin C-kieliseksi. Cfrontin tuottama ohjelmakoodi käännettin tämän jälkeen tavallisella C-kääntäjällä. Pian C++ sai kuitenkin myös suoran kääntäjän. 1983 kielen nimi muuttui C with Classes-kielestä C++:aan. Kieli sai samalla uusia ominaisuuksia, kuten esimerkiksi operaattorien ja funktioiden nimien ylikuormitukset, virtuaaliset funktiot ja uuden kahden kauttaviivan kommenttimerkinnän. 1989 ilmestyi kielen 2.0-versio, joka toi uusina ominaisuuksina mm. moniperinnän, abstraktit luokat ja staattiset jäsenfunktiot. Seuraavana vuonna julkaistu *The Annotated C++ Reference Manual* määritti C++-kielelle suunnan. Se oli myös kielen standardoinnin lähtökohta. Esimerkki C++ -kielestä on kuvassa 5. Kuvan ohjelma tulostaa Hello World-tekstin Windows-käyttöjärjestelmän ikkunaan.

Yhteensopivuuden säilyttäminen C-kieleen oli C++ -kielelle kaksiteräinen miekka. Toisaalta se mahdollisti olemassa olevan C-kielisen lähdekoodin uudelleenkäytön ja lisäsi siten kielen potentiaalista käyttäjäkuntaa. Yhteensopivuuden säilyttämisen vuoksi jouduttiin kuitenkin tekemään kompromisseja, jotka johtivat puhtaan oliopohjaisen ajattelun rikkoutumiseen [Sut99].

C++ -kielestä puuttuu myös roskienkeruu, joka on olio-ohjelmointikielille tyypillinen ominaisuus. Tosin sen jättäminen kielestä pois oli Stroustrupilta tietoinen ratkaisu. Vaikka roskienkeruu helpottaa ohjelmoijan työtä, se voi aiheuttaa ongelmia aikakriittisissä sovelluksissa. C++ -kielelle on kuitenkin olemassa lisäkirjastoja, jotka lisäävät kieleen roskienkeruuominaisuudet.

```
#include <afxwin.h>

class CHello : public CFrameWnd {
public:
    CHello() {
        Create(NULL,_T("Hello World!"),WS_OVERLAPPEDWINDOW,rectDefault);
    }
};

class CHelloApp : public CWinApp {
public:
    virtual BOOL InitInstance();
};

BOOL CHelloApp::InitInstance() {
    m_pMainWnd = new CHello();
    m_pMainWnd->ShowWindow(m_nCmdShow);
    m_pMainWnd->UpdateWindow();
    return TRUE;
}

CHelloApp theApp;
```

Kuva 5: Windows -käyttöjärjestelmän graafista käyttöliittymää hyödyntävä C++ -kielellä laadittu Hello World -ohjelma

C++ -kieli voi hyvin ja se kehittyy yhä. Sen ensimmäinen virallinen standardoitu versio ilmestyi 1998 ja korjattu versio vuonna 2003. Viimeisimmät uudet standardiin ehdotetut ominaisuudet ovat vuodelta 2005.

4.2 Internet-aikakauden oliokieli: Java

Jos C-kieli ja Simula olivat esikuvia C++ -kielelle, niin Java-kieli otti paljon vaikutteita C++ -kielestä. Myös vähemmän tunnettu C-kielen variantti, Objective-C, on ollut Java-kielen ideoiden lähteitä [BC05].

Java-kielen ja suoritusalustan kehitys alkoi vuonna 1991, jolloin Sun Microsystemsillä työskennelleet James Naughton, Mike Sheridan ja James Gosling alkoivat työskennellä salaisen The Green Project-työn parissa. Projektin tarkoituksena oli kehittää työkaluja uudenlaisten sulautettujen järjestelmien ohjelmointiin.

Kehitystyön alkuvaiheessa James Gosling otti C++ -kielen työkalujensa perustaksi, mutta alkoi parannella sitä omien tarpeidensa mukaan. Alkuperäinen C++ -kieli koettiin liian monimutkaiseksi, virhealttiiksi ja resurssisyöpöksi. Kieli ei myöskään ollut kaikilta osin riittävän alustariippumaton. Gosling kehittämää C++ -johdannaista kutsuttiin nimellä C++ ++ --. Pian kehitystiimi päätti kuitenkin hylätä C++ -kielen muokkaamisen. He aloittivat kokonaan uuden alustan ja ohjelmointikielen kehittämisen. Gosling antoi uudelle kielelle nimeksi Oak, hänen työhuoneestaan näkyneen puun mukaan [Oco95].

Javan edeltäjän, Oakin, ensimmäinen versio valmistui loppukesästä 1992, jolloin sen ominaisuuksia esiteltiin Star7-kämmentietokonella. Vaikka Star7-laitetta ei koskaan julkaistu, se riitti vakuuttamaan yleisön Oak-alustan käyttökelpoisuudesta, ja sen ansiosta Oak julkaistiin Sunin ulkopuolelle. Julkaisun jälkeen Oakia tarjottiin mm. kaapeli-tv-yhtiöille lähetysten vastaanottamiseen käytettyjen päätelaitteiden ohjelmointia varten [Bou98].

Kuva 6: Star7 -kämmentietokone
[http://java.sun.com/features/1998/05/images/star7.small.jpg]

Oak ei koskaan saavuttanut suosiota kaapelipäätelaitteissa, joten 1994 Oak suunnattiin web-järjestelmien ohjelmointiin. 1994 internet oli vielä monille tuntematon käsite, mutta Oakin kehittäjäjoukko uskoi sen suosion kasvuun. Oak oli ensimmäisiä tekniikoita, jotka mahdollistivat web-sivujen muokkaamisen interaktiivisiksi. Oakin etuina oli myös alustariippumattomuus hyvin heterogeenisessä internetin käyttäjäkunnassa. Esitelläkseen Oakin webiin soveltuvia ominasuuksia Naughton ohjelmoi WebRunner -nimisen Web-selaimen, joka myöhemmin nimettiin uudelleen HotJavaksi. 1994 Oak -alustalle annettiin tavaramerkkiongelmien vuoksi uusi nimi: Java.

Suurelle yleisölle Java esiteltiin vuonna 1995, jolloin julkaistiin Javan ensimmäinen versio. Kielen lanseerauksen yhteydessä silloinen selainmarkkinoiden ylivoimainen johtaja, Netscape, ilmoitti sisällyttävänsä selaimiinsa Java-tuen [Bou98].

Javaa lähdettiin heti alussa kehittämään moderniksi oliokieleksi. Javan suunnittelun lähtökohtina oli alustariippumattomuus, käyttämisen helppous, turvallisuus ja suora tuki tietoverkkojen hyödyntämiseen. Kieleen haluttiin myös koota aiempien olio-ohjelmointikielten hyvät puolet [BC05]. Alustariippumattomuus Javassa toteutetaan virtuaalikoneen avulla. Java-kielinen lähdekoodi käännetään välikielelle, jota kutsutaan tavukoodiksi. Tavukoodi suoritetaan Javan virtuaalikoneen avulla. Sen vuoksi Javan tavukoodiksi käännetyt ohjelmat toimivat periaatteessa missä tahansa järjestelmässä, jolle on laadittu tavukoodia suorittava virtuaalikone. C++ -kielestä poiketen Javassa on myös automaattinen roskienkeruu. C-kielten ongelmana on ollut juuri muistin käyttämisen vaikeus. Ohjelmoija on joutunut itse huolehtimaan muistin varauksesta ja vapautuksesta. Javan virtuaalikoneen roskienkerääjä huolehtii täysin käyttäjilleen näkymättömästi muistin vapautuksesta, jolloin muistinvarauksesta johtuvat virheet poistuvat. Roskienkerääjä ja virtuaalikone ovat olleet keskeisiä tekniikoita useissa olio-ohjelmointikielissä, kuten Simulassa ja Smalltalkissa.

Vaikka C++ ja Java ovat syntaktisesti hyvin lähellä toisiaan, niin Javaa yksinkertaistettiin sekä rakenteellisesti että käsitteellisestikin. Javasta puuttuu joitain tehokkaita, mutta virhealttiita toimintoja, kuten goto-komento tai osoittimet ja suorat muistiviittaukset. Java ei myöskään sisällä monia muita monimutkaisuutta aiheuttaneita ominaisuuksia, kuten operaattorien ylikuormittamista tai moninperintää. Kuvassa 7 esitellään Javan syntaksia. Kuvan ohjelma tulostaa näytölle tekstiä.

```
public class HelloWorld {
   String msg;

   public HelloWorld() {
      this.msg = "Hello World!"';
   }

   public showHello() {
      System.out.println(this.msg);
   }

   public static void main(String[] args) {
      HelloWorld hw = new HelloWorld();
      hw.showHello();
   }
}
```

Kuva 7: Java-kielinen Hello World -sovellus, joka tulostaa Hello World -tekstin käyttäen hyväkseen Javan luokkaominaisuuksia

Vaikka Java suunniteltiin alunperin web-sovellusten toteuttamiseen, niin siellä sen on korvannut suositummat tekniikat, kuten Macromiedian Flash tai Shockwave. Myös Microsoftin kiinnostus Java-tekniikkaan laimeni Sunin estettyä Microsoftia tekemästä siihen omia epäyhteensopivia laajennuksiaan. Microsoft on sen jälkeen siirtynyt käyttämään syntaksiltaan ja luonteeltaan hyvin Javamaista C# -kieltä, vaikka tarjoaakin Java-kieleen tottuneille osittain yhteensopivan J# -kielen. Vaikka Javaa usein moititaan sen hitaudesta, on se silti laajasti käytössä sen siirrettävyyden vuoksi. Tunnetuimpia Java-alustalle tehtyjä sovelluksia ovat mm. Eclipse, Netbeans ja Azureus.

4.3 Muut olioperustaiset ohjelmointikielet

Olio-ohjelmointikielet eivät suinkaan rajoitu tässä tutkielmassa esiteltyihin kieliin, vaan niitä on ollut lukemattomia. Huomattavimpia muita olio-ohjelmointikieliä ovat mm. Objective-C, Eiffel, Oberon. Kuten luvussa 4.2 todettiin, Objective-C on ollut eräs Java-kielen esi-isistä. Eiffel puolestaan on puhtaasti oliopohjainen kieli, joka ei monista edistyneistä ominaisuuksistaan huolimatta ole saavuttanut laajaa suosiota.

Oliokielten suosion kasvun myötä moniin perinteisiin ohjelmointikieliin on lisätty oliopiirteitä. Mm. Basic-kielestä on tehty olioversioita, joista tunnetuin lienee Microsoftin Visual Basic. Myös suositulle Pascalille on nykyaikainen oliolaajennus Delphi. Monet muutkin suositut skriptikielet, kuten PHP ja Perl, ovat saaneet olio-ominaisuuksia, vaikka ne ovat olleet usein osittain puutteellisia. Esimerkiksi PHP sai monet keskeiset oliopiirteet, kuten kapseloinnin, abstraktit luokat ja konstruktorit sekä destruktorit vasta versiossa 5.0, joka ilmestyi kesällä 2004.

Eräs merkittävä uusi oliokieli on Microsoftin .NET ympäristöön perustuva C#, joka on hyvin Java-kielen kaltainen. C# -lähdekoodi käännetään Javan tapaan välikielle, jota .NET alusta suorittaa. Välikieltä kutsutaan nimellä Common Language Runtime. C#-kielen käyttö ei rajoitu vain Microsoft Windows -alustalle, sillä .NET ympäristön käyttämä välikieli standardoitiin vuonna 2005. .NET ympäristöstä on ilmestynyt monia Microsoftista riippumattomia toteutuksia, kuten esimerkiksi Mono tai DotGNU, jotka perustuvat vapaaseen lähdekoodiin ja ovat käyttöjärjestelmäriippumattomia [Wik05c].

5 Yhteenveto

Olio-ohjelmointikielet syntyvaihe ajoittuu 1960-luvun alkuun, jolloin kehitettiin ensimmäinen olio-ohjelmointikieli Simula. Vaikkei Simula saavuttanutkaan suurta kaupallista menestystä, sen ansiot olio-ohjelmointikielten esivanhempana ovat kiistattomat. Simulasta sai alkunsa Smalltalk, joka on myös merkittävä oliokielten kehittäjä. Simula on ollut esikuva monille nykyisillekin oliokielille, kuten C++:lle.

C++ -kielen ja Javan myötä olioperustaiset kielet ovat otettu käyttöön yleisesti. Ennen lähinnä akateemisiin piireihin rajoittuneesta ohjelmointiajattelusta tuli C++- ja Java-kielten ansiosta yleisesti tunnettua, ja nykyään paradigma on saavuttanut johtoaseman erilaisten ohjelmointityylien joukossa.

Olio-ohjelmointikielet ovat tulleet suosituiksi niiden huomattavien etujen vuoksi. Perinteisiin kieliin verrattuna ne helpottavat uudelleenkäyttöä mm. periytymisen ja modulaarisuutensa vuoksi sekä vähentävät virhealttiutta kapseloinnin vuoksi. Oliopohjainen ajattelun avulla voidaan myös ohjelmointiongelmat jakaa ihmismielelle loogisempiin kokonaisuuksiin. Nykyisten valtakielten Javan, C++:n ja C#:n suosiosta voidaan päätellä olioperustaisuuden olevan tehokas ja käyttökelpoinen ajattelumalli. Olio-ohjelmointi on tullut jäädäkseen ainakin ennen seuraavaa suurta ohjelmoinnin murrosta.

Lähteet

BC05 Baeumle-Courth, P., An Introduction to Java for C++ Programmers, 2005. URL `http://info.baeumle.com/java/intro/philosophy.html`. (9.2.2006).

Bou98 Bouys, J., Java technology: The early years, 1998. URL `http://java.sun.com/features/1998/05/birthday.html`. (23.1.2006).

Cap03 Capretz, L. F., A brief history of the object-oriented approach. *SIGSOFT Softw. Eng. Notes*, 28,2(2003), sivu 6.

DaN05 Dahl, O.-J. ja Nygaard, K., How object-oriented programming started, 2005. URL `http://heim.ifi.uio.no/~kristen/FORSKNINGSDOK_MAPPE/F_OO_start.html`. (19.1.2006).

Har97 Harold, E., The comp.lang.java FAQ List, 1997. URL `http://www.ibiblio.org/javafaq/javafaq.html`. (9.2.2006).

Kay93 Kay, A. C., The early history of smalltalk. *HOPL-II: The second ACM SIGPLAN conference on History of programming languages*, New York, NY, USA, 1993, ACM Press, sivut 69–95.

Oco95 O'Connell, M., Java: The inside story, 1995. URL `http://sunsite.uakom.sk/sunworldonline/swol-07-1995/swol-07-java.html`. (9.2.2006).

Ren82 Rentsch, T., Object oriented programming. *SIGPLAN Not.*, 17,9(1982), sivut 51–57.

Soe05 Soegaard, M., Object orientation redefined: From abstract to direct objects and toward a more understandable approach to understanding, 2005. URL `http://www.interaction-design.org/mads/articles/object_orientation_redefined.html`. (9.2.2006).

Str93 Stroustrup, B., A history of C++: 1979-1991. *HOPL-II: The second ACM SIGPLAN conference on History of programming languages*, New York, NY, USA, 1993, ACM Press, sivut 271–297.

Sut99 Sutherland, J., A history of object-oriented programming languages and their impact on program design and software development, 1999. URL `http://jeffsutherland.com/papers/Rans/OOlanguages.pdf`. (19.1.2006).

Tun98 Tuntematon, Basic Aspects of Squeak and the Smalltalk-80 Programming Language, 1998. URL `http://www.cosc.canterbury.ac.nz/~wolfgang/cosc205/smalltalk1.html`. (19.1.2006).

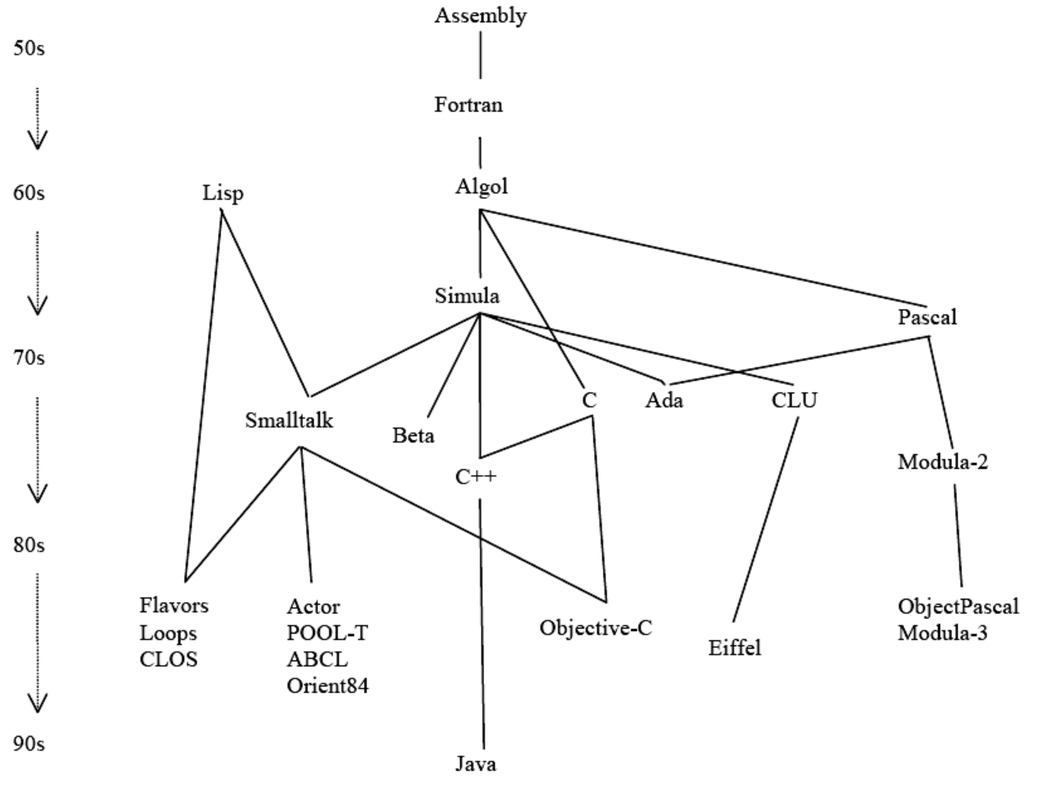
Wik98 Wikla, A., *Ohjelmoinnin perusteet Java-kielellä*. OtaDATA, Espoo, 1999.

Wik05 Wikipedia, Microsoft_.NET — Wikipedia, the free encyclopedia, 2005. URL `http://en.wikipedia.org/wiki/Microsoft\_.NET`. (23.1.2006).

Wil96 Wileden, J., Programming languages and software engineering: past, present and future. *ACM Comput. Surv.*, 28,4es(1996).

Liite 1. Kaavio oliokielten kehityksestä

Kaavio kuvaa oliokielten kehitystä. Kaaviossa on mukana myös kieliä, jotka eivät ole olio-ohjelmointikieliä, mutta ovat olleet keskeisiä vaikuttajia oliopohjaisten kielten kehittymiseen.



Kuva 8: Olio-ohjelmointikielten kehitys [Cap03]