Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Ohjelmoinnin suuntautumisvaihtoehto
Nikolaos Mäenpää

Opinnäytetyö

Maksuttomat oppimisympäristöt Java-opiskelun tukena

Työn ohjaaja Paula Hietala (FL)
Työn tilaaja Tampereen Ammattikorkeakoulu,
Tietojenkäsittelyn koulutusohjelma,
Pekka Pöyry (FM)
Tampere 09/2010

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma, Ohjelmistotuotanto

Tekijä	Nikolaos Mäenpää
Työn nimi	Maksuttomat oppimisympäristöt Java-opiskelun tukena
Sivumäärä	32
Valmistumisaika	Syyskuu 2010
Työn ohjaaja	Paula Hietala
Työn tilaaja	Tampereen ammattikorkeakoulu, tietojenkäsittelyn koulutusohjelma

Tiivistelmä

Tässä opinnäytetyössä käsitellään maksuttomia Java-oppimisympäristöjä. Tavoitteena on arvioida ympäristöjä erityisesti aloittelevan ohjelmoinnin opiskelijan kannalta ja tutkia niiden soveltuvuutta ohjelmoinnin alkeiden opiskelun tueksi. Työ on tehty Tampereen ammattikorkeakoulun tietojenkäsittelyn koulutusohjelman toimeksiantona. Työ toimii sekä opettajan apuna tämän valitessa mahdollisia apuvälineitä opetuksen tueksi, että opiskelijan oppaana tämän etsiessä oppimistaan helpottavia työkaluja.

Työtä varten vertailtiin kolmea eri oppimisympäristöä – Javalaa, BlueJ:ta ja Alicea. Vertailtuja osa-alueita olivat mm. asennus, käyttöliittymä, tarjolla oleva ohjedokumentaatio sekä ohjelman havainnollisuus ja hyödyllisyys aloittelevalle ohjelmoinnin opiskelijalle. Ennen ympäristöjen esittelyä työssä käydään läpi ohjelmoinnin peruskäsitteitä ja niiden oppimisen ongelmakohtia. Lopuksi esitetään ajatuksia siitä, miten ohjelmia voisi käyttää ohjelmoinnin ensimmäisen kurssin kehittämisessä TAMK:ssa.

Avainsanat Java, ohjelmointi, oppimisympäristö, Javala, BlueJ, Alice

TAMK University of Applied Sciences
Business Information Systems, Software Development

Writer	Nikolaos Mäenpää
Thesis	Improving the Understanding of Java through the Use of Free Learning Environments
Pages	32
Graduation time	September 2010
Thesis Supervisor	Paula Hietala
Customer	Tampere Polytechnic University, Degree Programme in Business Information Systems

Abstract

This thesis discusses free Java learning environments. The objective is to evaluate the environments especially from a novice programmer's point of view and examine their suitability for supporting the studying of the basics of computer programming. The thesis is an assignment from Business Information Systems department of Tampere Polytechnic University (TAMK). It works both as an aid for teachers who are choosing supporting instruments for teaching and as a guide for students who are looking for tools to improve their learning.

Three different learning environments were tested for this thesis – Javala, BlueJ and Alice. The qualities that were compared consisted of e.g. installation, user interface, available help documentation and environments' illustrativeness and usefulness for a novice computer programming student. Before demonstrating the environments, the thesis will scrutinize some of the basic concepts of computer programming and the problems associated with learning them. Finally, there will be some thoughts about how the environments could be used for improving the first computer programming course in TAMK.

Keywords Java, programming, learning environment, Javala, BlueJ, Alice

Sisällysluettelo

1 Johdanto

Tietokoneohjelmoinnin opiskelu aloitetaan nykyään useimmiten Java-kieltä käyttäen. Java on oliopohjainen ohjelmointikieli, joka on saavuttanut suuren suosion ja levinnyt erittäin laajalle aina matkapuhelimista palvelinsovelluksiin asti. Ohjelmoinnin opiskelun Javalla aloittavan on opittava samalla kertaa sekä ohjelmoinnin yleinen logiikka, että korkean abstraktiotason ohjelmointikielen käyttö. Korkea abstraktiotaso sinällään kyllä helpottaa ohjelmoijan työtä, mutta punaisen langan löytäminen oppimisen alussa voi olla monelle hankalaa.

Olio-ohjelmoinnin sekä Javan oppimisen avuksi on kehitetty erilaisia oppimisympäristöjä, joilla pyritään havainnollistamaan ohjelmakoodin rakennetta ja toimintaa. Tässä työssä tutustun joihinkin näistä ympäristöistä ja pyrin vertailemaan niitä kattavasti muun muassa havainnollisuuden, käytön helppouden ja saatavilla olevan dokumentaation osalta. Ympäristöjen ominaisuudet, kohderyhmät ja lähestymistavat vaihtelevat, ja tämän työn tavoitteena on tuottaa hyvin eritelty vertailu, jonka pohjalta opiskelija tai opettaja voi löytää tarpeisiinsa soveltuvan ohjelmoinnin oppimisympäristön.

Työtä varten testattiin kolmea erilaista oppimisympäristöä: Javala, BlueJ ja Alice. Ohjelmien lukumäärä rajattiin melko pieneksi, jotta vältyttäisiin turhalta luettelomaisuudelta työn rakenteessa. Testattavaksi valitut ohjelmat ovat keskenään hyvin erilaisia, eikä niitä voi asettaa paremmuusjärjestykseen. Työn tarkoitus onkin toimia yleisenä esittelynä eikä ohjelmille anneta arvosanoja.

Ohjelmoinnin opettelusta ja ohjelmoinnin alkeista on kirjoitettu kirjoja hyllymetreittäin, ja tällaisia kirjoja onkin käytetty lähteenä työn alkuosan yleistä osiota kirjoitettaessa. Oppimisympäristöistä ei kuitenkaan ole olemassa juurikaan kirjallisuutta, joten pääasiallinen lähdeaineisto työlleni on itse ympäristöt ja niitä käyttämällä ja tutkimalla saatu tieto. Lisäksi apuna on käytetty internetistä löytyvää aineistoa kuten ohjedokumentaatiota, arvosteluja ja esittelyjä.

Työn toimeksiantaja on Tampereen ammattikorkeakoulun tietojenkäsittelyn koulutusohjelma. TAMK on opiskelijamäärältään Suomen kolmanneksi suurin ammattikorkeakoulu, ja tietojenkäsittelyn opinnot aloittaa koulussa vuosittain yli sata opiskelijaa. Työn kirjoittaja työskentelee ohjelmointialalla ja on tutustunut aloittelevien opiskelijoiden kohtaamiin ongelmiin paitsi aloittaessaan omia opintojaan, myös pitäessään ohjelmoinnin tukiopetustunteja TAMK:ssa.

Ohjelmat testattiin Intel Core2Duo-prosessorilla ja kahden gigatavun muistilla varustetulla Asus-tietokoneella. Käyttöjärjestelminä olivat Ubuntu Linux 8.04, jossa BlueJ-ohjelman testaaminen suoritettiin, sekä Microsoft Windows Vista Home Premium, joka toimi Alicen testiympäristönä. Javalaa testattiin molemmissa käyttöjärjestelmissä Mozilla Firefox 3 -selaimella.

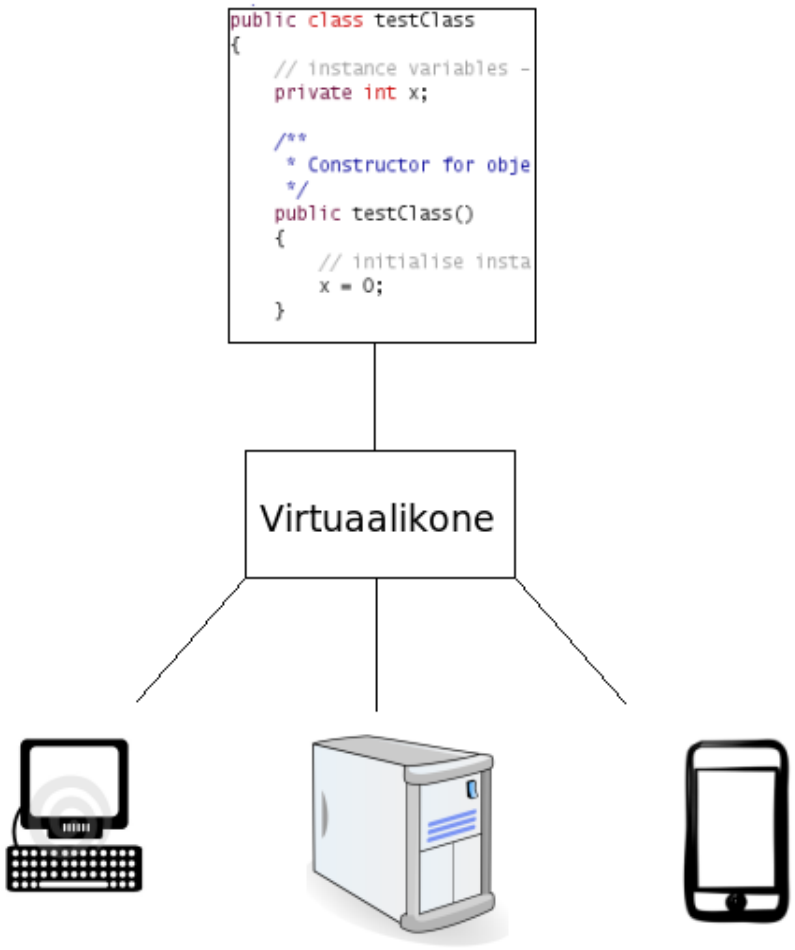
Työn alussa käydään läpi ohjelmoinnin opiskelun aloittamista yleisellä tasolla, sekä siihen liittyviä ongelmakohtia. Tämän yhteydessä työssä tullaan selkeyden vuoksi kuvailemaan joitain ohjelmoinnin avainkäsitteitä, jotta myös ohjelmointia osaamaton lukija voisi paremmin ymmärtää tekstiä. Oppimisympäristöjä käsittelevä osuus alkaa yleiskuvauksella testatuista ympäristöistä ja samalla käydään hieman läpi niiden taustaa. Tämän jälkeen arvioidaan käyttöliittymää ja käyttöönoton helppoutta sekä saatavilla olevaa dokumentaatiota ja ohjemateriaalia. Lopuksi arvioidaan ympäristöjen käyttökelpoisuutta ja hyödyllisyyttä ja sitä, millaisille käyttäjille ohjelmat soveltuvat.

2 Lähtökohdat

2.1 Java

Moni ohjelmointia opiskeleva saa ensi kosketuksensa ohjelmointiin Java-kielen kautta. Java on Sun Microsystemsin 1990-luvulla kehittämä alustariippumaton oliopohjainen ohjelmointikieli. Sen isänä pidetään James Goslingia (hän on myös kehittänyt suositun Emacs-editorin). Se sai alkunsa Oak-nimisestä kielestä, joka kehitettiin PDA-laitteiden (kämmentietokone) ohjelmointia varten. Tarkoituksena oli kehittää kieli, joka olisi yksinkertainen ohjelmoida, turvallinen ja alustariippumaton (Niemeyer, Krudsen 2005, 3). Nykyään Javaa käytetään kaikenlaisissa kuviteltavissa olevissa laitteissa älykorteista suuriin palvelimiin.

Javan suurin etu, alustariippumattomuus, on Java-virtuaalikoneen ansiota - ohjelma ei kommunikoi suoraan käyttöjärjestelmän tai prosessorin kanssa, vaan välissä on ohjelmallinen “tietokone”, virtuaalikone. Se tulkkaa käskyt käyttöjärjestelmälle, ja ohjelmoijan ei tarvitse tietää kuin se, mitä käskyjä virtuaalikone ymmärtää (kuva 1). Tällainen tulkkaaminen syö jonkin verran resursseja, joten Javalla toteutettu ohjelma on monessa tapauksessa hitaampi kuin muulla kielellä toteutettu ja käyttöjärjestelmän ymmärtämään muotoon käännetty sovellus - tosin Java-ohjelmien hitaus on vähentynyt huomattavasti kielen alkuajoista. Sun Microsystems tekee myös erityisiä Java-prosessoreita, joilla Javaa voidaan käyttää ilman virtuaalikonetta.



Kuva 1: Virtuaalikone mahdollistaa saman koodin toimimisen monella alustalla

Java on monikäyttöisyytensä ja helppoutensa johdosta levinnyt erittäin laajalle ja sikäli se onkin looginen valinta ensimmäiseksi ohjelmointikieleksi. Ohjelmointia aiemmin tuntematon opiskelija voi kuitenkin pitää Javaa (tai oliopohjaisia kieliä yleensäkin) hieman hankalana ensikurkistuksena ohjelmoinnin maailmaan. Vaikka oliopohjaisuus helpottaakin ohjelmoijan työtä, ohjelman ylläpitoa ja jatkokehitystä merkittävästi, sen ymmärtäminen on monelle aluksi vaikeaa, kun ohjelmoinnin "kielioppi" ja ajatusmallit ovat muutenkin vielä epäselviä. Toisaalta oliopohjaisen ohjelmointiajattelun oppiminen alusta alkaen voi auttaa saamaan todellisen hyödyn irti olioista, kun pohjalla ei ole ei-oliokieliin liittyviä ajatusmalleja, jotka tulisi poisoppia (Barker, 2005, 4).

2.2 Ohjelmoinnin oppiminen

Ohjelmointikieliä voi joiltain osin verrata luonnollisiin kieliin. Vaikka ne periaatteessa ovatkin huomattavasti yksinkertaisempia ja yksiselitteisempiä, vie ohjelmoinninkin opettelu vuosikausia. Kuitenkin, jos osaa yhtä ohjelmointikieltä hyvin, oppii muitakin suhteellisen helposti, sillä perusperiaatteet kielten taustalla ovat samanlaiset. Siispä näiden perusperiaatteiden oppiminen onkin kaikki kaikessa. Vaikka useimmat tietokoneohjelmat ovat liki käsittämättömän monimutkaisia, ne koostuvat lopulta melko yksinkertaisista palikoista ja toiminnoista, joista muutamia käymme lyhyesti läpi seuraavaksi.

2.2.1 Ohjelmoinnin peruskäsitteet

Muuttuja

Muuttuja on “ohjelman muistipaikka, johon voidaan tallentaa yksittäinen tieto” (Harju & Juslin 2006, 9). Sitä voidaan ajatella lokerona, johon mahtuu yksi määrätyntyyppinen asia, kuten merkkijono tai numero. Javassa muuttujan tietotyyppejä ovat mm. String (merkkijono – oikeastaan String ei ole varsinainen tietotyyppi vaan luokka, mutta sitä voidaan käsitellä tietotyyppinä), int (kokonaisluku) ja double (desimaaliluku). Muuttujia havainnollistetaan tarkemmin luvussa 2.2.2 luokkia ja olioita käsittelevässä osiossa.

Toisto- ja ehtolauseet

Ohjelman kulku määräytyy aina joidenkin annettujen ehtojen puitteissa. Usein on myös suoritettava samaa tai samankaltaista komentosarjaa useita kertoja, kuten esimerkiksi tilausrivien kokonaissummaa laskettaessa. Ehtolauseet voivat olla esimerkiksi tyyppiä

```
jos ehto,
        tee asia a.
muuten tee asia b.
```

Toistolauseissa voidaan pyörittää tiettyä koodinpätkää joko ennalta määrätty määrä kertoja, tai kunnes annettu ehto toteutuu. Esimerkki:

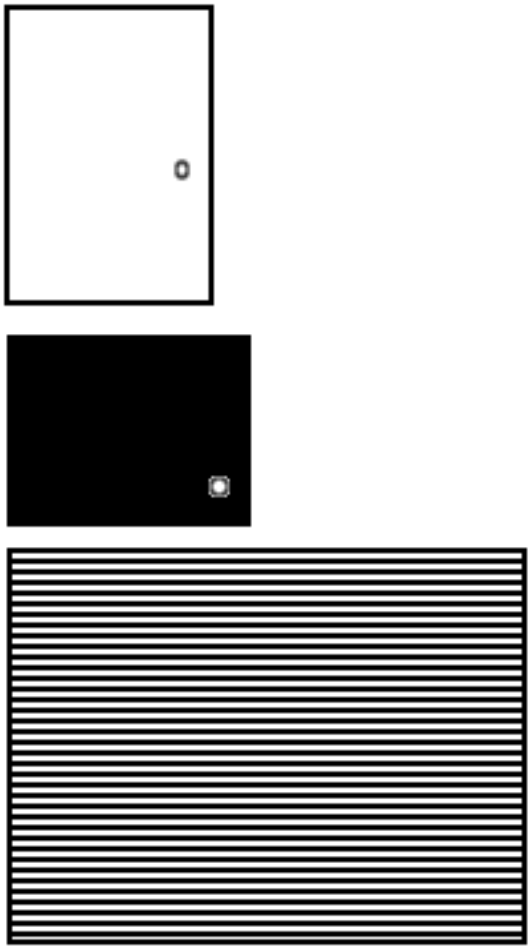
```
niin kauan kun taulukossa on soluja
        tulosta seuraavan solun sisältö.
```

2.2.2 Olio-ohjelmointi

Olio-ohjelmoinnissa ohjelma koostuu keskenään kommunikoivista olioista (object). Oliopohjainen rakenne on helppo hahmottaa ja ylläpitää, minkä johdosta oliokielet ovatkin nousseet suureen suosioon. Oliopohjaisia kieliä ovat mm. Java, C# ja C++. Tässä alaluvussa kuvaillaan lyhyesti olio-ohjelmoinnin keskeisimmät käsitteet: luokka, olio ja metodi.

Luokka ja olio

Oliokielissä olioiden piirteet määritellään tavallisesti olion luokassa (Koskimies, 2000, 34). Luokka määrittää, mitä ominaisuuksia ja toimintoja oliolla voi olla. Luokkaa voidaan ajatella jonkin asian yleisenä ideamaailman abstraktina kuvauksena, kun taas olio on sen reaalimaailman ilmentymä. Ihminen hahmottaa ja luokittelee maailmaa abstraktioiden kautta, ja se on hyödyllistä samoista syistä sekä ohjelmoinnissa että maailman havainnoinnissa. Periaatteessa olioajattelu on käsitteellisen ajattelun vastine ohjelmointimaailmassa. Esimerkiksi kaikki maailman ovet voidaan ajatella "ovi-luokan" ilmentyminä, eli niillä on tietyt ominaisuudet (muuttujat), joita muuttamalla saamme kuviteltua erilaisia ovia (kuva 2). Tiedämme esimerkiksi, että ovella on ominaisuuksia kuten korkeus, leveys ja väri, sekä toimintoja, kuten avaudu ja sulkeudu.



Kuva 2: Ovet

Kuvassa esimerkkinä ulko-ovi, kaapin ovi ja autotallin ovi.

Ulko-ovi: korkeus 200 cm, leveys 90 cm, väri: valkoinen.

Kaapin ovi: korkeus 50 cm, leveys 70 cm, väri: musta.

Autotallin ovi: korkeus 200 cm, leveys 250 cm, väri: raidallinen.

Metodi

Metodi on olion osaama toiminto, tyypillisesti joko olion tilaa muuttava, tilaa tutkiva tai olion osille jonkin tietyn toimenpiteen tekevä toiminto (Koskimies 2001, 37). Ovi-oliolla voi olla esimerkiksi toiminnot avaudu, sulkeudu, lukitse ja avaa lukitus, sekä ti -laa tutkivat toiminnot kerro leveys, kerro korkeus, kerro väri (sekä olion tilaa eli omi-naisuuksia muuttavat toiminnot muuta leveys, muuta korkeus, muuta väri).

Muodostaja

Muodostaja eli konstruktori on metodi, jota kutsutaan oliota luotaessa. Sen avulla voidaan asettaa joko kiinteitä tai parametreina annettavia alkuarvoja olion muuttujille. Java-kielessä luokalla voi olla yksi tai useampia muodostajametodeita (vain yhtä voidaan kutsua kulloistakin oliota luotaessa).

JavaDoc

Ohjelmakoodin kommentointi on tärkeä osa ohjelmointityötä ja elintärkeää ohjelman ylläpidon ja jatkokehityksen kannalta. Jopa ohjelmoijan itsensä voi olla vaikea ymmärtää kirjoittamaansa koodia jälkeenpäin, jos sitä ei ole kommentoitu kunnolla. Moniin ohjelmointikieliin on kehitetty apuvälineitä kommentoinnin selkeyttämiseen, ja JavaDoc on yksi tällaisista työkaluista.

Kun ohjelmoija kommentoi koodinsa JavaDoc-standardin mukaisesti, voidaan lähdekoodista luoda JavaDoc-tiedosto. Asianmukaiseen kommentointiin kuuluu muun muassa jokaisen luokan ja metodin vakiomuotoinen kommentointi muuttujien esittelyineen. Kunnolla ja standardien mukaisesti kommentoidusta tiedostosta syntyy JavaDocilla html-tiedosto, jossa on selkeästi eriteltynä ohjelman eri elementit, kuten metodit parametreineen ja palautusarvoineen. Tällaisen dokumentaation avulla ohjelmoijan on helppo tutustua ohjelmaan kun hän jatkokehittää sitä.

3 Testatut ympäristöt - yleiskuvaus

Oppimisympäristöllä voidaan käsittää monenlaisia asioita aina sosiaalisesta ympäristöstä opiskelutiloihin, mutta tässä työssä sanaa käytetään vain ohjelmista, jotka on tarkoitettu ohjelmoinnin oppimiseen. Normaali ohjelmistokehitys ja usein ohjelmoinnin opetuskin tapahtuu kehitysympäristössä, jonka tarkoitus on taata työkalut täyspainoiseen ohjelmointiin. Kehitysympäristö on kuitenkin suunniteltu ammattilaisen tarpeita silmälläpitäen mahdollistamaan tehokas työskentely, kun taas aloittelija saattaa kaivata hieman pelkistetympää lähestymistapaa. Esimerkiksi Tampereen ammattikorkeakoulussa opetuksessa käytettävä Sun Microsystemsin Eclipse-kehitysympäristö on kyllä monipuolinen ja hyvä työkalu, mutta juuri monipuolisuutensa tähden myös ensivaikutelmaltaan sekava.

Testattujen oppimisympäristöjen tarkoitus on tarjota matalan kynnyksen opettelumahdollisuus ohjelmointiin. Käyttäjän ei tarvitse osata asentaa ja konfiguroida hankalia kehitysympäristöjä kuten vaikkapa Eclipseä, vaan hän voi keskittyä suoraan itse ohjelmoinnin opetteluun. Jos opiskelu aloitetaan asentamalla monimutkaisia ympäristöjä ja säätämällä lukuisia asetuksia, voi opiskelijan keskittyminen kohdistua aluksi ohjelmoinnin kannalta toissijaisten ongelmien kanssa painimiseen.

Vertailluista ympäristöistä osa asennetaan omalle koneelle ja osa toimii suoraan verkkoselaimessa. Selainpohjaista ympäristöä voidaan pitää kaikkein matalimman aloituskynnyksen omaavana, sillä mitään asennusta tai konfigurointia ei tarvitse suorittaa ennen käytön aloittamista. Myös alustariippumattomuus on selainpohjaisuuden etuja.

3.1 Javala

Javala on suomalainen Timo Lehtosen kehittämä selaimessa toimiva Java-oppimisympäristö. Se on kehitetty Tampereen teknillisen yliopiston ohjelmistotekniikan laitoksella ja ensimmäinen versio julkaistiin vuonna 2004. Sovellus on tehty Javalla ja se pohjautuu avoimen lähdekoodin komponentteihin.

Javalan käyttökieli on suomi, joten heikommallakin kielitaidolla varustettu käyttäjä pääsee oppimiseen kiinni helposti. Javalan käyttö edellyttää, että ohjelmoinnista tietää jo jotain eli liikkeelle ei lähdetä aivan alkeista - peruskäsitteet pitäisi olla valmiiksi ainakin auttavasti hallussa, jotta Javalasta saa hyödyn irti. Javalan sivullakin kerrotaan, että käyttäjän oletetaan osaavan valmiiksi ohjelmoinnin perustaidot esimerkiksi C++-kielellä, ja myös olio-ajattelun tulisi olla tuttua. Käytännössä helpoimpia harjoituksia voi kuitenkin tehdä tyhjästä aloittanut Java-opiskelijakin jo ensimmäisten ohjelmoinnin oppituntiensa jälkeen.

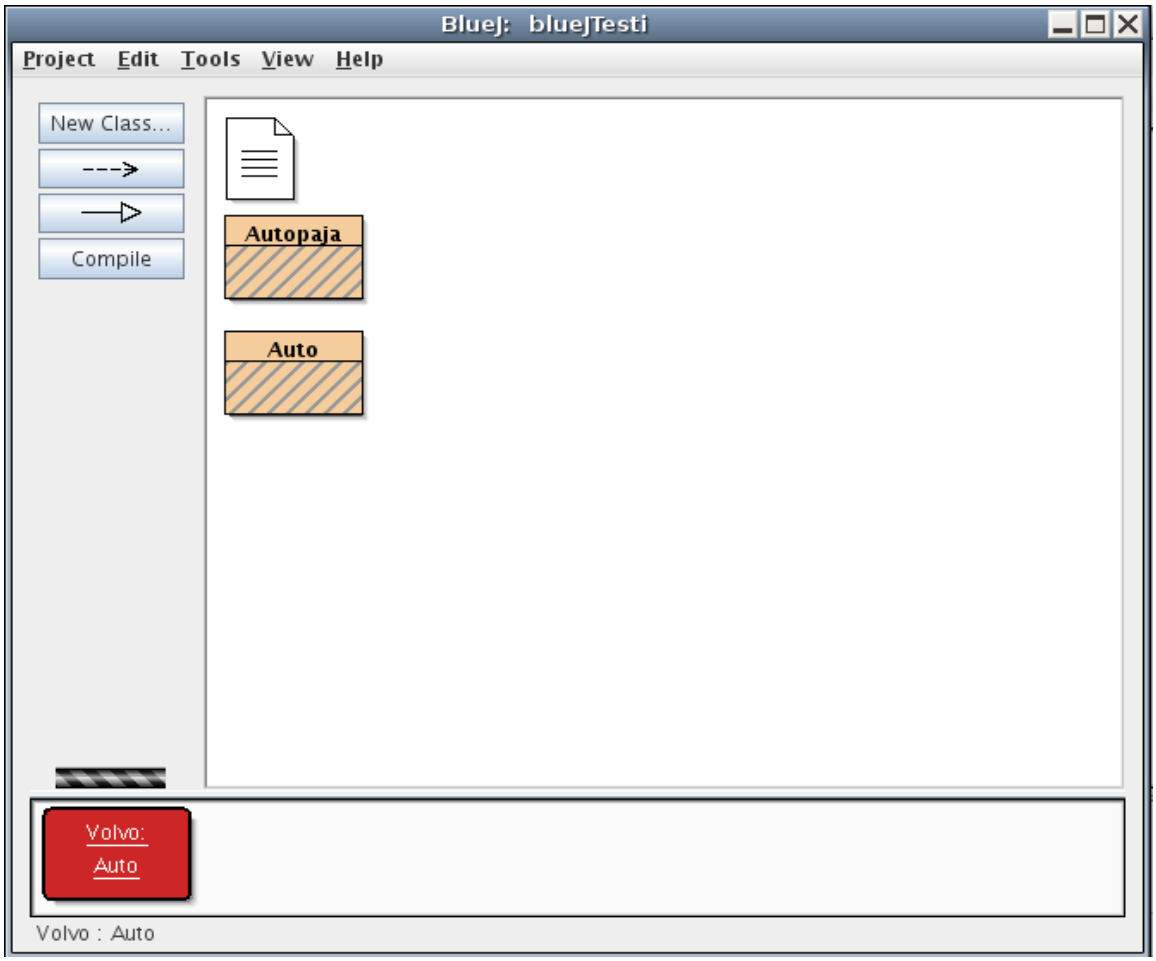
Javalan sisältö on jaettu lyhyisiin luentoihin aihealueittain, alkaen merkkijonoista, olioista ja luokista aina I/O-käsittelyyn ja grafiikkaan asti (kuva 3). Jokaisesta aiheesta on tarjolla hieman teoriaa, koodiesimerkkejä ja harjoituksia. Harjoituksia tekemällä voi kerätä pisteitä rekisteröimälleen nimimerkille ja päästä sivuston top-listalle. Tämä ominaisuus saattaa hyvinkin motivoida käyttäjän tekemään enemmän harjoituksia ja näin saavuttamaan paremman rutiinin.

Kuva 3: Javalan aloitusikkuna. Vasemmalla on osa-aluevalikko ja oikeanpuoleiseen osioon aukeaa teksti ja esimerkit. Harjoitukset aukeavat omaan ikkunaansa.

3.2 BlueJ

BlueJ on australialaista alkuperää oleva oppimisympäristö. Se on seuraaja Michael Köllingin ja John Rosenbergin kehittämälle Blue-oppimisympäristölle, jossa ohjelmointia opeteltiin ympäristön omalla erityisellä kielellä. BlueJ:ssä oppiminen tapahtuu Javalla. Ohjelma on käyttäjän omalle koneelle asennettava alustariippumaton Java-ohjelma. Windowsille, MacOS X:lle ja Debian-pohjaisille Linuxeille on toteutettu omat asennuspaketit, mutta ohjelmaa voi Java-pohjaisuutensa takia käyttää periaatteessa missä tahansa ympäristössä.

BlueJ on ainoa vertailuun otettu ohjelma, joka on myös “täysiverinen” kehitysympäristö. Normaaleista kehitysympäristöistä se poikkeaa siten, että se on erityisesti suunniteltu aloitteleville ohjelmoinnin opiskelijoille. Luokat ja niiden väliset suhteet ovat helposti tarkasteltavissa päänäytöllä, ja luokista voidaan luoda olioita ja näiden metodeja voidaan kutsua helposti muutamalla hiirenpainalluksella (kuva 4). Verrattuna esimerkiksi Eclipseen BlueJ on toiminnoiltaan erittäin riisuttu ja täten selkeämpi, mutta ominaisuuksien vähyyden vuoksi sillä ei voi käytännössä tehdä mitään pieniä harjoitusohjelmia monimutkaisempia kokonaisuuksia. Käyttökielenä BlueJ:ssä on englanti.



Kuva 4: BlueJ:n käyttöliittymä. Luokat, niiden mahdolliset suhteet ja readme-tiedosto näkyvät päänäkymässä. Luokista voidaan tehdä olioita alapalkkiin tarkempaa havainnointia varten.

3.3 Alice

Alice on testatuista ympäristöistä omaperäisin - ideana on ollut tehdä näyttävä 3d-ympäristö, jonka avulla havainnollistetaan ohjelmoinnin peruskäsitteitä. Kohderyhmänä Alicen sivuilla kerrotaan ohjelmoinnin ensi askeleita ottavat peruskoulusta yliopistoon, ja sitä käytetään ohjelmoinnin opetuksessa yli 10 prosentissa yhdysvaltalaisista collegeista.

Alice-projekti sai alkunsa jo 1990-luvun alussa, ja ohjelman kehittäjinä on toiminut tutkijoita, opettajia ja opiskelijoita useista amerikkalaisista yliopistoista. Aluksi kyseessä ei ollut ohjelmoinnin oppimisympäristö, vaan Alicea kehitettiin työkaluksi varhaisten virtuaalitodellisuusympäristöjen luomista varten. Nykyään projektia rahoittavat mm. peliyhtiö Electronic Arts, Microsoft, Google ja Disney. Kehitteillä olevassa Alicen kolmannessa versiossa tullaan näkemään hahmoja Electronic Artsin Sims 2 -pelistä.

Ohjelman käyttäjä ei tarvitse lainkaan aikaisempaa tietämystä ohjelmoinnista, ja "koodaus" hoidetaan drag and drop -menetelmällä havainnollisesti ja helposti. Varsinaisesta ohjelmakoodin syntaksista ei siis tarvitse tietää mitään - eikä sitä Alicen avulla myöskään opi. Pienenä miinuksena ohjelma toimii vain Mac OS- ja Windows-käyttöjärjestelmissä.

Alice on periaatteessa tietokoneanimaatiostudio, jossa käyttäjä oppii olio-ohjelmoinnin käsitteitä helposti ja havainnollisesti. Käyttäjä aloittaa valitsemalla halutun "maailman", ja lisäilee sinne objekteja (olioita) laajasta valikoimasta. Oliolla on erilaisia ominaisuuksia, kuten koko ja väri, sekä metodeja, kuten liiku, käänny, soita ääni ja muuta kokoa (kuva 5). Toiminnot voidaan määrittää käynnistymään tiettyjen ehtojen täyttyessä, tai automaattisesti ajastettuna. Alicella voi jopa luoda yksinkertaisia pelejä, ja valmiiden esimerkkien joukosta löytyykin mm. lento"simulaattori" (kts. Liite 1).

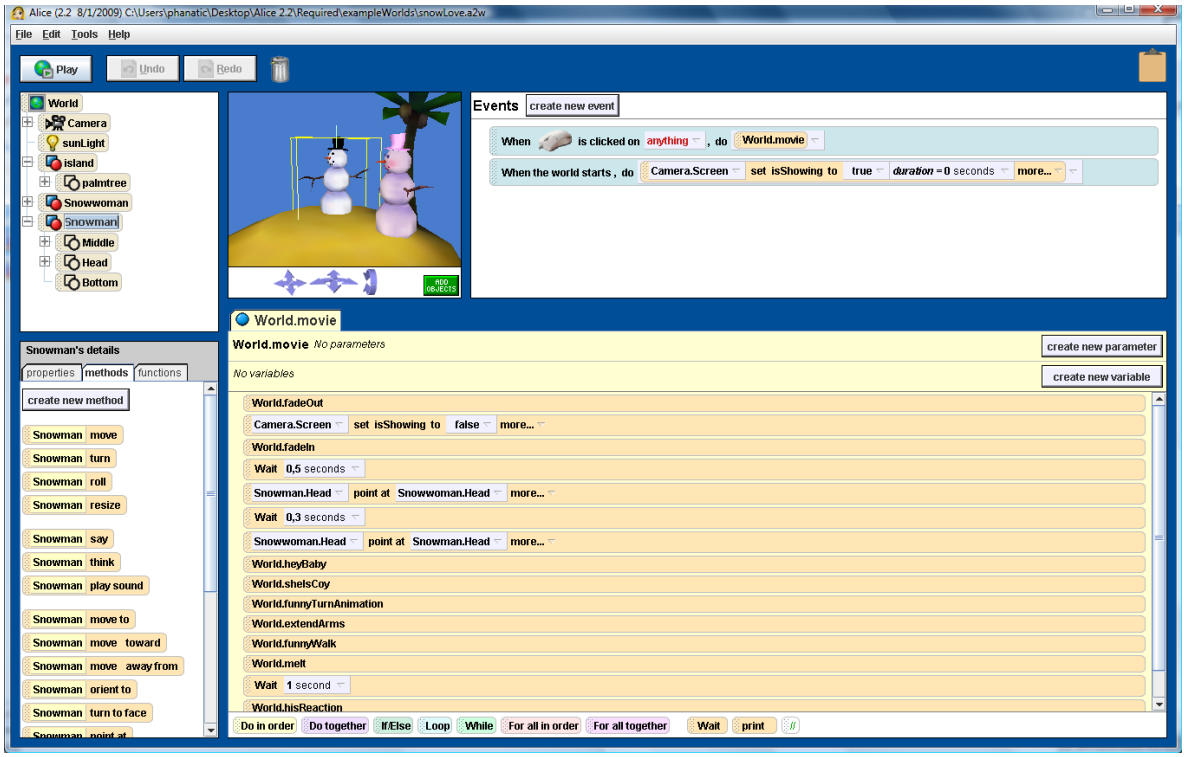
Ilmentymämuuttujat, ehto- ja toistolauseet ja vertailuoperaattorit tulevat tutuksi nopeasti. Ensivaikutelma ohjelmasta on selkeä, ja tutoriaalia seuraamalla lähes kuka tahansa saa ohjelmalla aikaiseksi ainakin yksinkertaisen animaation. Toiminnot hoituvat

drag and drop -periaatteella suhteellisen intuitiivisesti - tosin täysin ilman manuaalia ei tätäkään ohjelmaa käytetä. Selkeän kuoren alla on kuitenkin valtavan monipuolinen ympäristö, ja kun ominaisuuksiin alkaa kunnolla tutustumaan, saattaa helppouden tunnekin jäädä taka-alalle. Ohjelmointi ei ole yksinkertaista, eikä siitä voi sellaista tehdä edes nätin graafisen oppimisympäristön avulla. Jos käyttäjä haluaa tehdä monipuolisen ja hienon animaation, riittää ehto- ja toistolauseissa, kameran siirtelyssä ja muussa säätämisessä tekemistä yökausiksi.

Käyttäjä voi esimerkiksi luoda auton, määritellä siihen liittyville muuttujille (kuten koko, väri) arvot ja sijoittaa auton tielle. Vasemman yläkulman oliolistauksesta valitaan auto-olio, ja tällöin vasempaan alakulmaan aukeaa nähtäväksi kaikki lista auton ominaisuuksista ja "taidoista". Auto voi mm. liikkua eteen ja taakse, kääntyä ja äännellä (töötätä). Näitä toimintoja voidaan sitten asettaa suoritettavaksi halutussa järjestyksessä ja haluttujen ehtojen puitteissa.

Helppoudesta huolimatta Alicessa siis riittää syvyyttä paljon enemmän kuin ensi näkemältä arvaisi. Päiväkoti-ikäisetkin voisivat käyttää ohjelman perustoimintoja yksinkertaisten animaatioiden luomiseen (mikäli englannin kieli on hallussa kohtalaisesti), mutta jopa ohjelmointia työkseen tekevä kirjoittaja huomasi uppoutuvansa tuntikausiksi hiomaan pienen testianimaationsa yksityiskohtia.

Suurin etu Alicessa on ohjelman esitystapa. Normaalisti ohjelmoinnin opiskelussa tehdään alkulukulaskureita ja muita merkkipohjaisia ohjelmia, ja useimmille muutaman numeron tulostuminen ruutuun ei välttämättä motivoi opiskelemaan asiaa lisää. Alicella saa tehtyä hienon animaation voltteja heittävästä ilmalaivasta lyhyemmässä ajassa kuin mikä menisi painoindeksilaskurin ohjelmoimiseen Eclipsellä.



Kuva 5: Alicen käyttöliittymä. Ylhäällä vasemmalla ovat maailmassa olevat objektit (oliot), ja vasemmassa alalaidassa voidaan tarkastella niiden toimintoja (metodeja). Ylhäällä keskellä on näkymä itse maailmasta, sen oikealla puolella interaktiivisten tapahtumien luontinäyttö, ja alhaalla oikealla esimääritettyjen tapahtumien hallinta.

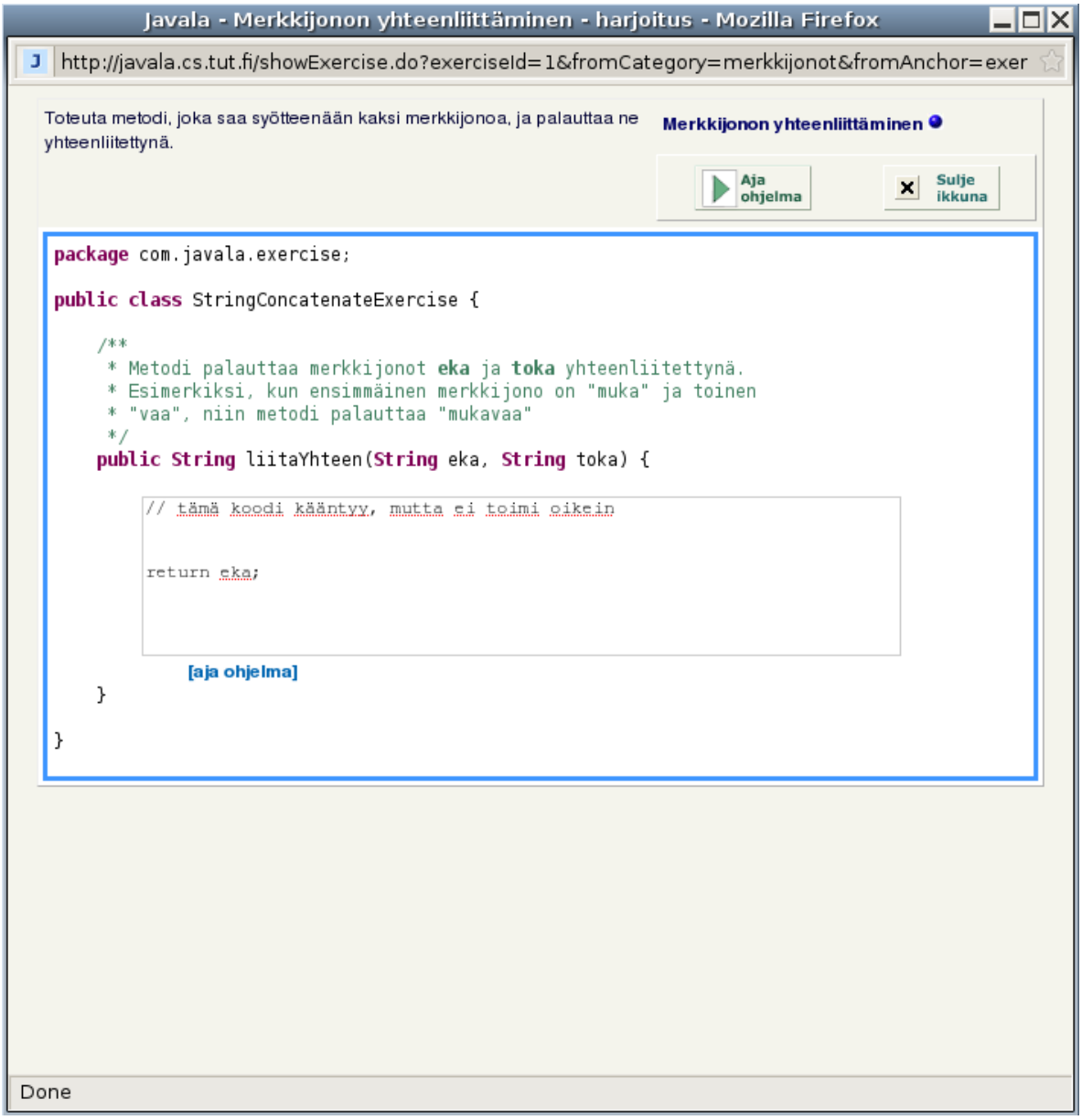
4 Asennus, käyttöönotto ja käyttöliittymä

Koska testatut ympäristöt on tarkoitettu nimen omaan uutta asiaa opetteleville opiskelijoille, on niiden käyttöönoton kynnys pyritty selvästi pitämään matalana. Voidaan perustellusti odottaa, ettei oppimisympäristöä opiskelunsa tueksi asentava henkilö halua aloittaa urakkaa lukemalla pitkiä ohjekirjoja, joten myös testitilanteessa ensikosketus ympäristöihin haettiin vailla merkittäviä ennakkotietoja. Testattujen ohjelmien käytön aloittamiseen riittää parhaimmillaan nettisivulle surffaaminen ja hankalimmillaankin opiskelijalta vaaditaan vain Java-ympäristön (Java Runtime Environment eli JRE) sekä itse oppimisympäristön asentaminen, jotka molemmat sujuvat muutamalla hiiren klikkauksella.

4.1 Javala

Javala ei vaadi minkäänlaista asennusta vaan toimii suoraan verkkoselaimessa. Käyttöönotto sujuu siis nopeasti, ainoa asia joka vaaditaan on käyttäjänimen rekisteröiminen. Sen avulla kerätään pisteitä harjoituksista. Käyttöliittymä on "nettisivumainen", vasemmasta reunasta valitaan haluttu aihealue ja ruudun keskiosaan tulee esimerkkejä ja teoriaa. Tekstin keskellä on linkkejä erilaisiin harjoituksiin jotka aukeavat omaan ikkunaansa (kuva 6).

Ohjelmaa testattiin etenemällä aihealueita järjestyksessä helpoimmasta alkaen ja tekemällä mielivaltaisesti valittuja harjoituksia kustakin aihealueesta, sekä myös valitsemalla muutamia kiinnostavia yksittäisiä lukuja ja niihin liittyviä tehtäviä. Täsmäopiskelu tiettyä aihealuetta kerratessa tai opetellessa on varmasti ohjelman todennäköisin käyttötapa, eikä Javalaa selvästikään ole suunniteltu alusta loppuun päntättäväksi paketiksi.



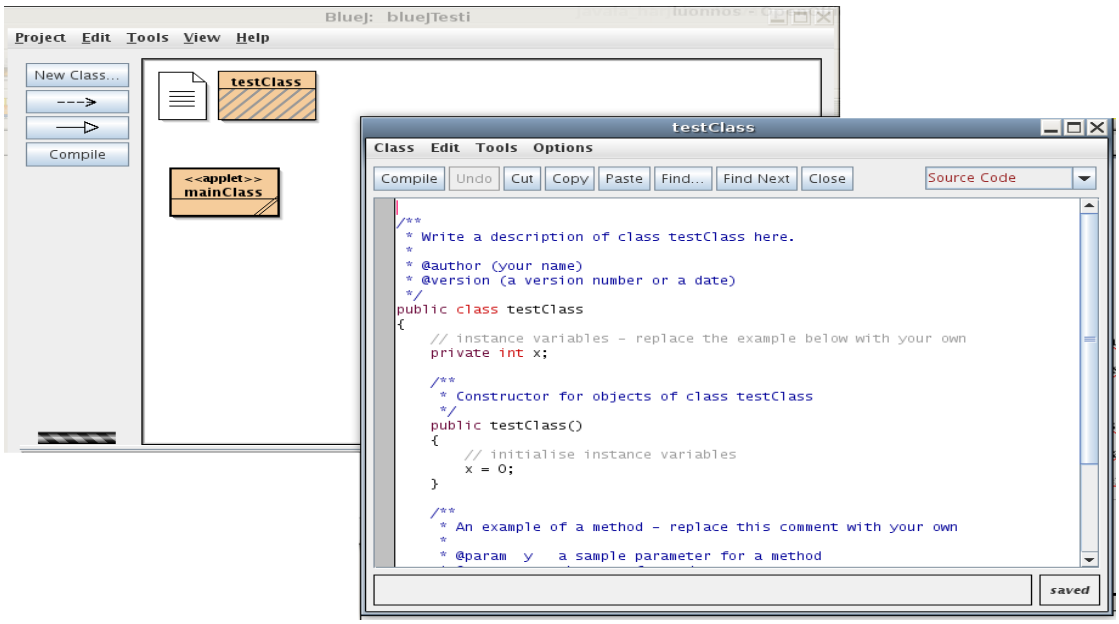
Kuva 6: Javalan harjoitusikkuna

4.2 BlueJ

BlueJ asennetaan käyttäjän koneelle, ja se on Java-ohjelma, jonka luvataan toimivan millä tahansa käyttöjärjestelmällä. Asennus sujuu vaivattomasti ja nopeasti.

Käyttöliittymä on hyvin yksinkertainen, mutta BlueJ:tä on silti mahdollista käyttää kokonaisten omien ohjelmien tekemiseen. Tekijöiden mukaan tavoitteena on ollut se, ettei käyttäjän tarvitse tuhlata aikaansa käyttöliittymän kanssa kamppailemiseen vaan he voivat keskittyä varsinaiseen ohjelmoinnin oppimiseen.

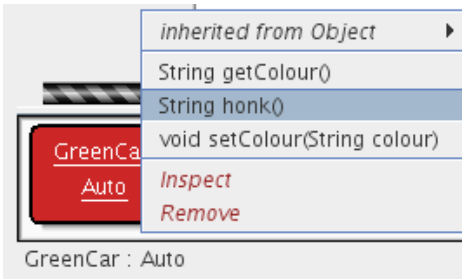
Perusnäkymässä on eräänlainen työpöytä, jolle projektin luokat tulevat näkyviin kuvakkeina. Varsinainen koodi kirjoitetaan erikseen aukeavissa luokkakohtaisissa ikkunoissa - täten käyttäjällä voi olla samanaikaisesti näkyvissä useita luokkia (kuva 7). Koodinäkymän sijaan luokan ikkunassa voidaan valita näkymään myös JavaDoc-dokumentaatio luokan sisällöstä. BlueJ:llä voi opetella myös unit-testausta (ohjelmakoodin toimivuuden testausta varta vasten suunniteltuilla automatisoiduilla testitapauksilla) ohjelmassa olevien työkalujen avulla.



Kuva 7: BlueJ:n käyttöliittymä on selkeä

Koska BlueJ on oikea, toimiva ohjelmointiympäristö, sitä testattiin luomalla yksinkertainen ohjelma. Tavoitteena oli havainnollistaa olio-ohjelmoinnin perusteita samankaltaisella tavalla kuin ensimmäisillä aiheen oppitunneillakin. Kirjoitettiin yksinkertainen Auto-luokka, jonka avulla pystyttiin luomaan erilaisia auto-olioita, joilla kullakin oli oma nimi ja väri, toiminnot värin muuttamiseksi ja tiedustelemiseksi, sekä tööttäys-toiminto (joka tulostaa ruudulle “Tööt”-tekstin).

Normaalisti oliot luodaan ohjelman ajamisen aikana, ja näin voi toimia myös BlueJ:ssä, mutta olio-käsitteen havainnollistamiseksi objekteja voidaan siinä manipuloida myös suoremmin. Käyttäjä voi napauttaa työpöydällä olevaa luokkaa hiiren kakkosnapilla ja valita uuden olion luomisen, jolloin ruudulle tulee ikkuna, johon annetaan olion nimi ja muodostajalle menevien parametrien arvot. Nyt olio on valmis ja sitä voidaan tarkastella ohjelman olionäytöllä. Sieltä käsin voidaan kutsua olion metodeita helposti hiiren painalluksella (kuva 8). Tällainen olioiden luonti ja manipulointi hiirellä helpottaa rakenteiden havainnollistamista verrattuna siihen, että kaikki tapahtuisi ohjelmakoodia kirjoittamalla. Opiskelijan ei tarvitse opetella tekemään kokonaista ohjelmaa, jotta saisi jotain konkreettista ruudulle, vaan luokkiin ja olioihin voidaan tutustua ja niitä voidaan oppia ymmärtämään “askel kerrallaan” ennen kuin niitä on osattava käyttää ohjelmassa.



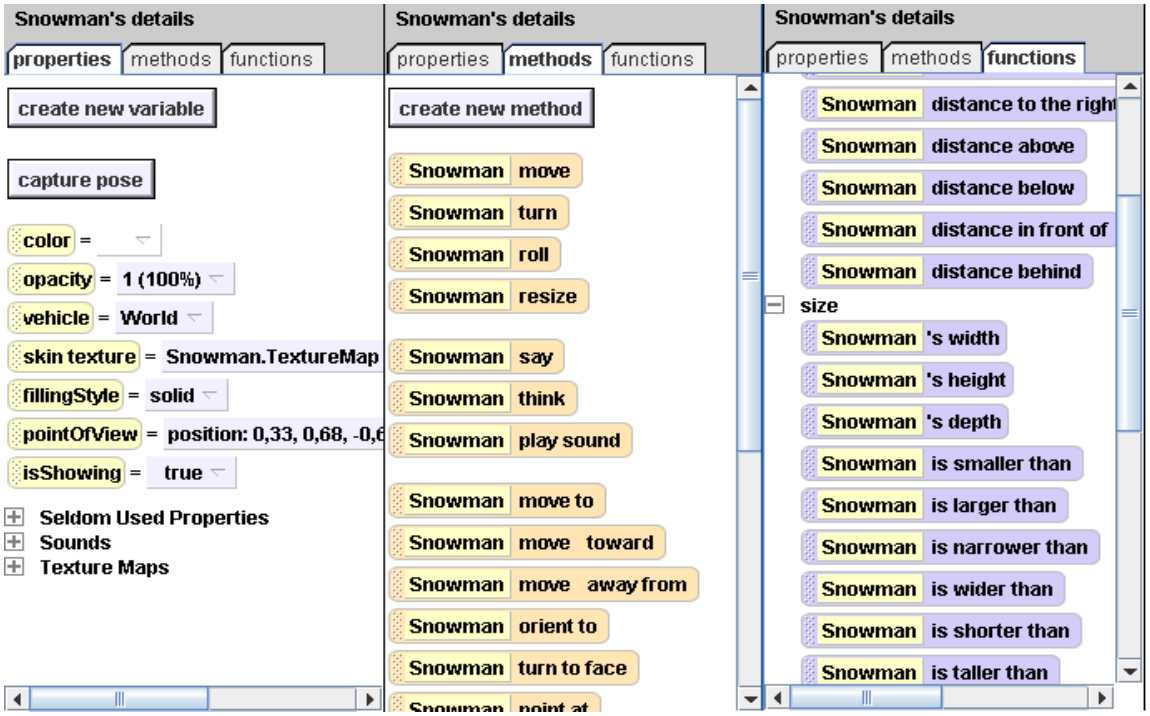
Kuva 8: Olion metodien kutsuminen BlueJ:ssa on toteutettu havainnollisesti

4.3 Alice

Alicen asennus on helppoa: riittää, että lataa ja purkaa paketin Alicen sivuilta, ja ohjelma on käyttövalmis. Ensimmäisellä käynnistyksellä käyttäjä viedään automaattisesti lyhyen tutoriaalin läpi, jossa esitellään ohjelman keskeisiä toimintoja ja käyttöliittymää. Noin viidessä minuutissa voi alkaa tekemään omia "ohjelmia" Alicella.

Käyttöliittymä vaikuttaa aluksi hieman sekavalta, mutta kun siihen pääsee sisälle, löytyvät useimmat toiminnot helposti. Jokainen ohjelma on Alicessa "maailma". Kaikkia siellä olevia objekteja voi tarkastella yläreunassa olevassa 3d-näkymässä, jonne uusia kohteita voi luoda mielensä mukaan. "Maailmassa" voi liikkua kameralla vapaasti, mutta hyvin kömpelösti.

Luonnollisesti ohjelman käyttöä aloittavan kannattaa aloittaa hyvin yksinkertaisesta maailmasta, jossa on vain yksi tai muutamia objekteja. Vasemmassa yläkulmassa on tarkasteltavana maailmassa olevat oliot, ja oliota napauttamalla sen tiedot aukeavat vasemmassa alakulmassa olevaan kehykseen. Siellä voidaan tarkastella olion muuttujia ja niiden arvoja, sekä sen osaamia toimintoja (kuva 9). Toiminnot on jaettu kahteen osaan, "metodit" ovat asioita, joita olio osaa tehdä ja "funktioilla" voidaan tarkastella ja muuttaa muuttujien arvoja (aksessorimetodit).



Kuva 9: Olion ominaisuuksien tarkastelua Alicessa

Alicen testauksessa oli tarkoituksena luoda yksinkertainen animaatio, jossa auto ajaa aavikolla kulkevaa tietä pitkin. Aluksi valittiin taustaksi aavikkomaisema, jonka jälkeen siirryttiin objektienlisäämistilaan. Tällöin näkymä maailmasta täyttää suuremman osan ruudusta, ja sinne voi valita erilaisia esineitä ja asioita, kuten ajoneuvoja, kasveja tai rakennuksia. Testitapauksessa aavikkomaailmaan lisättiin kaksi tienpätkää ja katuvaloa. Tienposkeen lisättiin myös kivenmurikka ja muutama kaktus. Lopuksi tielle asetettiin keltainen auto. Asioiden lisääminen maailmaan on melko kömpelöä, ja perspektiiviä pyöritettäessä ne vaikuttavat leijuvan maan yläpuolella. Alicen päätarkoitus ei kuitenkaan ole maailman vaan tapahtumien suunnittelu, joten tätä puutetta on turha korostaa liikaa. Kömpelyydestä huolimatta objektien manipuloimahdollisuudet ovat melko monipuoliset.

Kun maailma oli valmis, oli aika ryhtyä luomaan itse animaatiota. Autolle luotiin metodit "moveForward" (liiku eteenpäin), "moveBackward" (liiku taaksepäin), "turnLeft" (käänny vasemmalle) ja "turnRight" (käänny oikealle). Metodit liitettiin nuolinäppäinten painalluksiin, ja tuloksena oli pelkistetty autopeli, jossa käyttäjä voi ajaa autolla tietä (tai halutessaan myös tieltä). Autolle luotiin törmäystarkistusmetodi, jota

kutsutaan aina liikkumismetodien yhteydessä: jos havaitaan auton osuvan katuvalo-pylvääseen tai kiveen, se kimpoaa taaksepäin ja päästää kolahduksen.

Testikäytössä luotu “autopeli" havainnollistaa hyvin muuttujien, toisto- ja ehtolauseiden sekä metodien toiminnan. Monimutkaisemman maailman ja interaktioiden luominen olisi Alicella valtava urakka, mutta luonnollisesti se olisi tätä millä tahansa työkaluilla, "liikkuvien osien" määrä kasvaa helposti valtavaksi.

Näennäisestä yksinkertaisuudesta ja hieman lelumaisesta ulkoasusta huolimatta Alicen käyttöliittymästä paistaa läpi keskeneräisyys ja hiemankin monimutkaisempien asioiden tekeminen tuntuu raskaalta jatkuvan toimintojen etsimisen takia. Vika voi toki olla testaajassakin, mutta hieman yllättäen testatuista ympäristöistä juuri Alicen käyttössä tuntui olevan kaikkein eniten vaikeuksia. Osaltaan tämä johtunee siitä, että ohjelmointia jo osaavan testaajan ajatukset ovat "ohjelmoija-asennossa", kun taas Alice yrittää havainnollistaa ohjelmointia niille, jotka eivät vielä aihetta ymmärrä.

5 Ohjedokumentaatio ja muu oheismateriaali

5.1 Saatavuus

Vertailussa olleisiin oppimisympäristöihin liittyvän ohjedokumentaation saatavuus oli vaihtelevaa täysin ymmärrettävistä syistä.

Javala itsessään on periaatteessa interaktiivinen Java-ohjelmoinnin ohjekirja, eikä sen käyttöön tarvita enempää ohjeita kuin oppikirjan lukemiseen. Javalan käyttöönoton kynnys onkin vertailluista ohjelmista kaikkein matalin.

BlueJ:n omilta sivuilta löytyy PDF-muodossa kattavia manuaaleja eri tarpeisiin, aina asennuksesta unit-testaukseen asti. Sivuilta löytyy myös viitteet tieteellisiin julkaisuihin, joissa BlueJ:tä on käsitelty.

Alicessa on itsessään tutoriaalitoiminto, jolla uusi käyttäjä opetetaan ohjelman saloihin. Netistä löytyy melko paljon ohjeita monimutkaistenkin animaatioiden toteuttamiseen ohjelmalla ja onpa ohjelman käyttöä varten järjestetty jopa monipäiväisiä kursseja. Alicen saloihin opastavia kirjojakin on kirjoitettu, muun muassa Charles W. Herbertin An Introduction to Programming Using Alice.

5.2 Sisällön kattavuus

Javalan sivuilla ei ole varsinaisia ohjeita ympäristön käyttöön, lukuun ottamatta lyhyttä kuvausta eri osa-alueista sivun esittelyn yhteydessä. Ohjelman selkeyden ansiosta käyttäjä tuskin kaipaakaan tämän syvällisempää ohjeistusta.

BlueJ:n tekijöiden tarjoama dokumentaatio on erittäin kattavaa ja jaoteltu selkeästi eri pdf-tiedostoihin. Peruskäyttäjä voi halutessaan ladata asennus- ja käyttöohjeet, kun taas innostunut ohjelmoinnin harrastaja voi vaikka kehittää omia laajennuksiaan ohjelmaan tutustumalla dokumentaatioon blueJ:n APIa (ohjelmointirajapinta) koskien.

Alicen käyttöön löytyy netistä todella kattavasti ohjeita ja vinkkejä niin opiskelijoille kuin Alicea opetuksessa käyttäville opettajillekin. Esimerkiksi sivulla www.aliceprogramming.net (Dann, Cooper, Slater 2009) on materiaalia, jota opettaja voi käyttää kokonaisten kurssien opetuksen taustana ja apuna.

6 Arviota havainnollisuudesta ja hyödyllisyydestä

Testatut ohjelmat olivat kaikki havainnollisia ja hyödyllisiä apuvälineitä ohjelmoinnin opiskelun tueksi. Jokainen oli eri tarpeisiin suunniteltu, joten niitä on vaikea asettaa paremmuusjärjestykseen - kyse on käyttäjän omista vaatimuksista, mieltymyksistä ja taitotasosta.

Alicella lapsikin voi saada käsityksen olio-ohjelmoinnin peruskäsitteistä, mutta ohjelmoimaan sen avulla ei opi. Muut testatut ohjelmat eivät tuoneet niin havainnollisesti ja visuaalisesti esiin luokkia, olioita, metodeja sekä ehto- ja toistolauseita. Toisaalta Alicen käyttöliittymä on lopulta melko hankala ja epäintuitiivinen, ja ohjelman käyttäjä saakin varautua viettämään pitkät tovit tutoriaalien parissa.

Javala edellyttää käyttäjältään enemmän taitoja, mutta se on mainio apuväline perusasioiden kertaamiseen ja taitojensa testaamiseen. Opettaja voisi hyvin antaa opiskelijoilleen kotitehtäväksi Javalan tehtäviä, ja opiskelija taas palauttaa haluamansa asian mieleen selkeän aihealuejaon takia. Käyttöönotto sujuu erittäin nopeasti ja sivuston rakenne on selkeä.

BlueJ taas sopisi ohjelmoinnin ykköskurssin opetus- ja ohjelmointityökaluksi. Riisuttu käyttöliittymä ja ohjelman luomat luokkakaaviot vievät opiskelijan ajatukset olennaiseen.

Toimeksiantajaa kiinnosti, miten testattuja oppimisympäristöjä voisi hyödyntää ohjelmoinnin ensimmäisen kurssin kehittämiseksi. Mielestäni monimutkainen Eclipse-ympäristö, jota käyttäen opetus tällä hetkellä tapahtuu, muodostaa liian suuren kynnyksen oppimiselle. Ensimmäisellä kurssilla opeteltavat asiat voitaisiin hyvin toteuttaa BlueJ:llä, sillä yksinkertaisuudestaan huolimatta sillä onnistuu mm. tietokantayhteys JDBC-rajapinnan kautta – ominaisuus joka on tarpeen jo opiskelujen alkuvaiheessa. BlueJ:n tapa havainnollistaa luokkia ja olioita voisi estää monen opiskelijan lannistumisen uusien käsitteiden edessä ja näin auttaa ohjelmoinnin maailman avautumisessa yhä useammalle. Eclipse on ammattikäyttöön tarkoitettu työkalu ja valtaosa sen ominaisuuksista on turhia aloittelevalle ohjelmoinnin opiskelijalle. Toisaalta on hyvä, että

opiskelija tottuu “oikeisiin” työvälineisiin jo varhain, mutta mielestäni tässä tapauksessa Eclipsen haitat ovat hyötyjä suuremmat.

Javala on suositeltava väline itseopiskeluun ja tunneilla läpikäytyjen asioiden kertaamiseen. Myös esimerkiksi ohjelmoinnin kokeeseen valmistautuessa se on oiva työkalu. Opettajan olisikin hyvä ohjata opiskelijat Javalan sivuille – tosin näin taidetaan Tampereen ammattikorkeakoulussa menetellä jo nykyisinkin.

Alicella en näe suurta arvoa opiskelun tukena. Vaikka sitä käytetäänkin melko laajalti maailmalla ohjelmoinnin havainnollistamiseen, niin se on silti enemmänkin animaatiostudio ja ohjelmalelu kuin oikeasti hyödyllinen ohjelma. Toisaalta en luultavasti osaa nähdä Alicea samoin silmin kuin ohjelmointia osaamaton, joten lähestyn sitä eri näkökulmasta kuin aloitteleva opiskelija. Alicesta voisi olla lyhyen ohjelmoinnin opiskeluun valmistavan kurssin aiheeksi, mutta kuten työssä jo aiemmin todettiin, sen avulla ei opi ohjelmoimaan.

7 Lopuksi

Testatut ohjelmat olivat ominaisuuksiltaan varsin erilaisia ja eri käyttötarkoituksiin soveltuvia. Javala on enemmänkin kertaus- ja harjoitteluympäristö niille jotka jo osaavat Javaa jonkin verran, alkeiden opetteluun siitä ei mielestäni ole. Javalan avulla on helppo keskittyä tiettyyn yksittäiseen asiaan harjoitusten selkeän kategorisoinnin ansiosta. Alice taas on erinomainen apuvälien olio-ohjelmoinnin perusrakenteiden ja olio-maailman ymmärtämiseen, mutta koodin kirjoittamista se ei opeta, vaan on oikeastaan animaation skriptausohjelma eikä olio-ohjelmointympäristö. BlueJ on hyvä työkalu opetuksen apuvälineeksi ja huomattavasti esimerkiksi Eclipseä helpompana sopii mainiosti ensimmäisten ohjelmien kirjoittamiseen.

Ammattikorkeakoulutasoisessa opetuksessa BlueJ:n käyttö ohjelmointiympäristönä esim. Eclipsen sijaan, ja kotiharjoitukset Javalalla voisi olla hyvä yhdistelmä ohjelmoinnin opiskelun alkuun. Alice on kiinnostavuudestaan ja omaperäisyydestään ehkä liian kaukana oikeasta ohjelmoinnista ja jää todellisessa opiskelussa kuriositeetiksi. Sen sijaan lukion atk-tunneilla se voisi olla paikallaan.

Lähteet

Kai Koskimies, Oliokirja, 2009. Jyväskylä: Gummerus

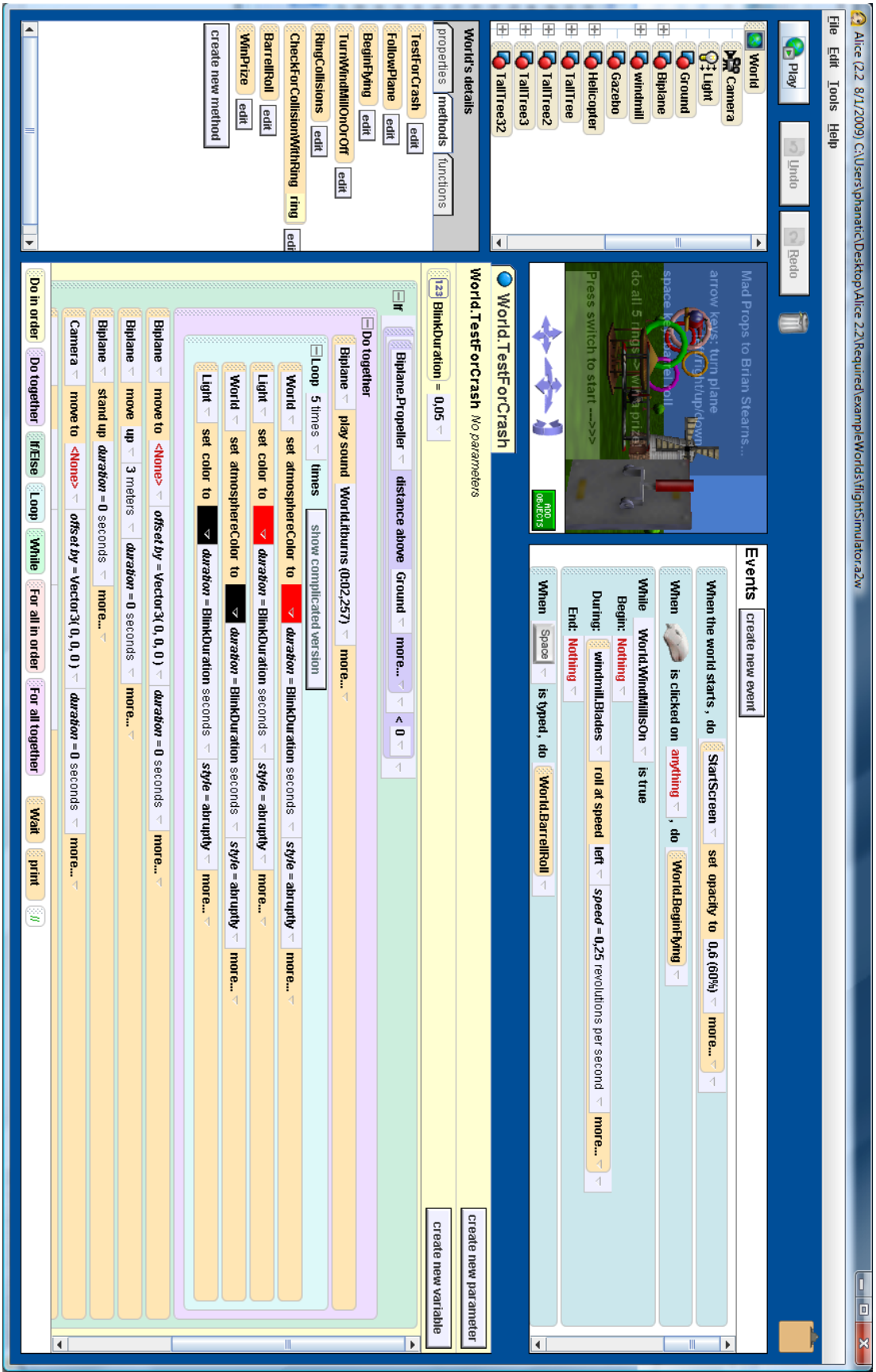
Jukka Harju ja Jukka Juslin, Tuloksellinen Java-ohjelmointi, 2006. Helsinki: Edita

Jacquine Barker, Beginning Java Objects - From Concepts to Code, 2005.USA: Apress

Patrick Niemeyer & Jonathan Krudsen, Learning Java, 2005. Sebastopol: O'Reilly

Dann, Cooper, Slater, Learning to Program with Alice, 2009
www.aliceprogrammers.net

Liitteet


Alice (2.2 8/1/2009) C:\Users\phanatic\Desktop\Alice 2.2\Required\exampleWorlds\flightSimulator.a2w
File Edit Tools Help
Play
Undo
Redo
World
Camera
Light
Ground
Biplane
windmill
Gazebo
Helicopter
TallTree
TallTree2
TallTree3
TallTree32
Mad Props to Brian Stearns...
arrow keys: turn plane
space: barrel roll
do all 5 rings -> win a prize
Press switch to start --->>>
600 OBJECTS
Events
create new event
When the world starts, do StartScreen set opacity to 0,6 (60%) more...
When is clicked on anything, do World.BeginFlying
While World.WindMillsOn is true
Begin: Nothing
During: windmill.Blades roll at speed left speed = 0,25 revolutions per second more...
End: Nothing
When Space is typed, do World.BarrelRoll
World.TestForCrash
World's details
properties methods functions
TestForCrash edit
FollowPlane edit
BeginFlying edit
TurnWindMillOnOrOff edit
RingCollisions edit
CheckForCollisionWithRing ring
BarrelRoll edit
WinPrize edit
create new method
World.TestForCrash No parameters
create new parameter
create new variable
BlinkDuration = 0,05
If Biplane.Propeller distance above Ground more... < 0
Do together
Biplane play sound World.itturns (0:02,257) more...
Loop 5 times times show complicated version
World set atmosphereColor to duration = BlinkDuration seconds style = abruptly more...
Light set color to duration = BlinkDuration seconds style = abruptly more...
World set atmosphereColor to duration = BlinkDuration seconds style = abruptly more...
Light set color to duration = BlinkDuration seconds style = abruptly more...
Biplane move to <None> offset by = Vector3(0, 0, 0) duration = 0 seconds more...
Biplane move up 3 meters duration = 0 seconds more...
Biplane stand up duration = 0 seconds more...
Camera move to <None> offset by = Vector3(0, 0, 0) duration = 0 seconds more...
Do in order
Do together
If/Else
Loop
While
For all in order
For all together
Wait
print
//