Ohjelmointi 1 / syksy 2007

6/20: Tietorakenteita

Paavo Nieminen

nieminen@jyu.fi

Tietotekniikan laitos

Informaatioteknologian tiedekunta

Jyväskylän yliopisto

Opelta lähti mopo käsistä?

Tilanne:

- Kentältä kuuluu ahdistunutta palautetta $\longrightarrow$ vakava tilanne.
- Demot olleet mahdottomia tehdä kurssilla toistaiseksi opitun perusteella $\longrightarrow$ erittäin vakava tilanne.

Missä todennäköisesti vika?

- Demotehtävät valmisteltu ja julkaistu ilman tietoa luentojen ja esimerkkien etenemisvauhdista.
- Unohtunut antaa ”se tärkein vinkki”.

Ongelmat, viat ja bugit ovat haasteita, jotka pitää yrittää korjata...

Mopedi haltuun

Annetaan ”se tärkein vinkki” välittömästi:

1. Tee vain yksi pieni muutos kerrallaan!
2. Jokaisen muutoksen jälkeen varmista että ohjelma toimii yhä!

   Eli käännä ohjelma ja suorita se; vertaa että tuloste on se, mitä oletit. Jatkuvasti!

$\longrightarrow$ näytetään käytännössä . . . hidastempoisesti . . .

Näin toimien on jopa vaikea päätyä tilanteeseen, jossa ohjelmasi ei toimisi lainkaan tai vaikuttaisi sekavalta vyyhdeltä sieltä täältä raavittuja langanpätkiä. Jos viimeisin lisäys ei toimi, tiedät mitä kysyä (ensin itseltä, sitten kaverilta, postilistalta, WWW:stä, oppikirjalta, vanhemmilta opiskelijoilta, opettajalta, . . .)

Kaino toive

Monella on motivaatio ottanut tarpeettoman iskun. Yritetään hidastaa tempoa ja vääntää lisää rautalankaa.

Toivottavasti huomiset näyttävät paremmalta ja motivaatio palautuu.

Kiitos palautteesta; ilman keskustelua ei voi syntyä tuloksia!

Erityinen hatunnosto sille, joka soitti huolistaan. Pieni näpäytys niille, jotka lähtivät suoraan virallisia teitä pitkin . . . ei se palaute sieltä pysty tulemaan luennoitsijan pääkoppaan asti niin että se ehtisi johtaa muutokseen tänään! Puhuminen auttaa; jupina on vain jupinaa.

Pedagoginen pohdinto: mihin mopo meni?

- Uskoisin, että tähän asti syntynyt tietotaso on jokaisella riittävä kolmen ohjelmointiviikon jälkeinen tilanne.
- Demo 3 säikäytti olemalla liian soveltava
- Toivottavasti ojassa on ainoastaan demo 3:n tehtäväsetti ja se että ”se tärkein vinkki” ei tullut riittävän selvästi ennen koodaustehtäviä.
- Tämä tilanne on uskoakseni paljon parempi kuin se, että ensin tsippailtaisiin kaksi kuukautta triviaaleilla tehtävillä ja havaittaisiin että pohja puuttuu, kun mitään ei enää ehtisi tekemään! Nyt ehditään!

Yritetään seuraava viikko tältä pohjalta. Jos ongelma ei poistu, tarkennetaan ongelmaa!

Kurssin tilanne ja suunta

- Tähän asti olennaista:
  - Ohjelmointi on lähdekoodin kirjoittamista (ja sitä edeltävää toimintaa)
  - Lähdekoodi jäsentyy eri tasoilla: mm. käännösyksikkö, luokka, metodi
  - Tarkimmalla tasolla lähdekoodi on lausekkeista koostuvia lauseita
  - Lauseke on operaattoreista ja operandeista koostuva lauseen osa, jolle määräytyy suorituksessa jonkuntyyppinen arvo (ellei lause ole void-tyyppisen metodin kutsu).
  - Toiminnallisuuden osia kannattaa paketoida aliohjelmiksi, joita voi kutsua monta kertaa mahdollisesti eri parametreilla.
- Tavoitteena on, että kaikki perusasiat on nähty kertaalleen huomisen luennon lopussa.
- Sitten kerrataan, havaitaan yksityiskohtia sekä taustoja ja opetellaan soveltamaan (yli puoli kurssia jäljellä tähän tarkoitukseen)

Tarkennusta: olion luonti ja viitemuuttuja

- Viite on jonkun olion ”osoite”, sen perusteella tietää, kenestä otuksesta puhutaan ja kenen sisäiseen tilaan luokan rajapinnan julkiset metodit operoivat.
- Viite (eli ”osoite”, ”osoitin”, ”tieto yksilön sijainnista”) on muuttuja siinä missä muutkin, joten sellaisen voi viedä parametrina aliohjelmalle! Aliohjelman kannalta viite toimii tällöin ihan samoin kuin aliohjelman sisällä esitellyt paikalliset viitteet. (Tästä aiheutuu realiteetti, että aliohjelma voi sivuvaikutuksenaan muuttaa parametrina osoitetun olion sisäistä tilaa! Asia vaatinee perusteellisen käsittelyn vielä myöhemmin, mutta sitä voi jo alkaa pohtimaan)
- Perustelluista syistä joistakin muuttujista (siis esim. viitemuuttujista) voi tehdä globaaleja.
- Ensimmäinen ohje: älä tee globaaleita muuttujia.
- Eräs toinen ohje: sitten kun tiedät, mitä teet (ja ettei siitä voi seurata ongelmia) niin joskus globaali muuttuja on hyödyllinen. Esim. Java-alustan tarjoama System.in on globaali, ja järkevä ollakin – kuvaahan se globaalia asiaa: standardisyötettä, joka on luettavissa koko ajan. Lienee siis kohtuullisen turvallista tehdä System.iniä lukevasta Scanner-luokan oliosta globaali, koska System.in on itsekin sellainen . . . Scanner vain käärii InputStreamin helppokäyttöisemmän rajapinnan taakse.

Tämän luennon rakenne

- Laajempia tietorakenteita: taulukko, merkkijono. (+ennakoiva huomio muista tietorakenteista).
- Taulukot Javassa: taulukko-oliot.
- Merkkijonot Javassa: luokkien String, StringBuilder (ja StringBuffer) oliot.
- Mitä se on kun on oliot: kerrataan luokan rajapinnan lukeminen ja käyttö, olion luominen ja metodien kutsuminen. (eli APIn soveltaminen).
- Taulukkoja sisältävät taulukot, esim. matriisi.
- Alustavasti algoritmeista taulukoiden ja merkkijonojen käsittelyssä.

Kun primitiivityypit eivät riitä

$\longrightarrow$ katsotaan esimerkkejä tarpeesta taulukolle: Tavoitteena tehdä sovellus, jolla tutkitaan opiskelijoiden kurssiin käyttämää aikaa ja joka varoittaa, jos liikutaan riskirajoilla, ts. 5% opiskelijoista käyttää yli 15 tuntia viikossa. Tutustutaan samalla tarkemmin parin asian soveltamiseen:

- Muuttuja, vakiomuuttuja
- for-silmukka, indeksimuuttuja
- Olion luonti dynaamisesti

Tarve merkkijonoille lienee selvä; ihminen kommunikoi kielellä, joka kirjoitettuna on jono merkkejä! Palataan merkkijonoihin taulukkoasian jälkeen . . .

Taulukko 1/4: perusidea

- Määritelmä: Taulukko on indeksoitu, määrätynkokoinen, kokoelma tietyn tyyppisiä muuttujia (Javassa siis primitiivejä tai tietyn luokan olioiden viitteitä).
- Taulukko on tyypillinen tietorakenne ohjelmointikielissä, ja sen syntaksikin on usein lähes samanlainen; opettelemme asian Javassa, mikä helpottaa saman idean käyttöä missä tahansa kielessä myöhemmin.
- Vahvasti tyypitetyissä kielissä (joissa siis muuttujien tyyppien suhteen ollaan jatkuvasti ”niuhotarkkoja”) myös taulukon tyyppi on määriteltävä tarkasti eikä se voi muuttua esittelynsä jälkeen.

Taulukko 2/4: syntaksi javassa

- Taulukon eli indeksoidun, määrätynkokoisen, kokoelman tietyn tyyppisistä muuttujista voi Javassa esitellä seuraavalla syntaksilla:

```
Tyyppi[] taulukonNimi;
```

- Kuten primitiiveille, taulukolle voi antaa esittelyn yhteydessä sisällön seuraavalla syntaksilla:

```
Tyyppi[] taulukonNimi =
   {arvo1, arvo2, ..., arvoN};
```

(Ero primitiivin alustukseen: arvoja on monta ja ne ovat kiharasulkujen sisällä)

- Alkioiden käyttö tapahtuu hakasulkusyntaksilla:

```
esimMuuttuja = esimTaulu[indeksi];
```

- Jos taulukkoa ei alusteta, se luodaan new:llä:

```
double[] neljanLuvunTaulu = new double[4];
```

Taulukko 3/4: Javassa taulukko on olio

- Javassa on huomattava, että taulukko on olio.
- Se käyttäytyy kaikistellen, kuten oliot aina.
- Taulukon tyyppi on siis olioluokka (vaikka sen käyttöön on erityinen syntaksi).
- Syntaksissa (ed. kalvo) `Tyyppi` on mikä tahansa tyyppi (primitiivi tai olioviite) ja taulukosta tulee viite indeksoituun, määrätynkokoiseen, kokoelmaan juuri sen tyyppisiä alkioita kuin mikä `Tyyppi` on.

Taulukko 4/4: useampia ulottuvuuksia

Alustava havainto moniulotteisista taulukoista:

- Taulukko voi sisältää taulukoita, esim. voidaan ajatella rivivektoria taulukkona koordinaateista, ja matriisia taulukkona rivivektoreista. Syntaksi kaksiulotteiselle taulukolle:

```
Tyyppi[][] taulukonNimi;
```

- Toinen esimerkki, jossa myös alustus:

```
double[][][] isoTaulu =
   { { {1.0, 2.0},
       {3.0, 4.0} },
     { {5.0, 6.0},
       {7.0, 8.0} } };
```

- Dimensioiden määrää rajoittaa vain se, mikä on järkevää.

Merkkijonot 1/4: Perusidea

- Yksi merkki on yksi merkki (esim. ’A’ tai ’b’ tai ’\’).
- Teksti koostuu useista merkeistä (esim. ”Arton ABC-kirja”).
- Sanotaan merkkien jonoa merkkijonoksi.
- Merkkijono vaatii tietokoneessa muistia vähintään niin paljon kuin sen merkit yhteensä.
- Merkkijonoja on kätevää käsitellä kuin ne olisivat kokonaisuuksia.
- Merkkijonoja käytetään todella moneen asiaan, mm. konsolisyöttöön, tulostukseen ja tiedon tallentamiseen.

Merkkijonot 2/4: Javassa merkkijono on olio

- Javassa on huomattava, että merkkijono on olio.
- Se käyttäytyy kaikistellen, kuten oliot aina. Erityispiirteinä mahdollisuus käsitellä merkkejä yksittäin ja osajonoina.
- Merkkijonon tyyppi on siis olioluokka, String (tai StringBuilder tai StringBuffer; ero ilmenee myöh. kalvolla).

Merkkijonot 3/4: esimerkkinä argumenttilista

→ Havaitaan tarkasti, mikä on String[] args, ja miten sitä käytetään: ohjelma, joka tulostaa argumenttinsa ja tutkii niistä ominaisuuksia kuten merkkijonon pituus, ensimmäinen ja viimeinen merkki ym. pikkuhassua. Samalla String-luokan APIa tutummaksi.

→ samalla kertautuu olioyksilön ja -viitteen ero: Parametrina mainiin tuleva muuttuja `args` on viite taulukko-olioon, josta on viitteitä merkkijono-olioihin.

Merkkijonot 4/4: Muuttumaton ja muuttuvainen olio

Merkkijonot ovat ensimmäinen kohtaaminen muuttuvaisten (mutable) ja muuttumattomien (immutable) olioiden kanssa:

- Esim. merkkijonoliteraalit ilmenevät suorituksen aikana String-luokan olioina. Niiden sisäinen tila ei koskaan voi muuttua olion elon aikana! Niiden luokan rajapinta ei mahdollista tätä.
- String-luokan merkkijono-operaatioissa syntyy aina uusi olio, jos tulos on erilainen kuin aiempi/aiemmat String-merkkijonot. Esim. yhteenliittäminen plussalla.
- Muuttuvainen merkkijono on tehtävä erikseen StringBuilder -luokan oliona (tai StringBuffer).

Java-kielen muuttujat ja tietoraken-teet

	Tyyppi	koko	literaaliesim.
Totuusarvot	boolean	?	true, false
Merkit	char	16	’A’, ’\u0041’
Kokonaisluvut	byte	8	14, 123, -128, 0x7A
	short	16	-14, 0123, 32767
	int	32	14, -1234, 32767
	long	64	14L, 92233720368547758O7
Liukuluvut	float	32	14.0, 1.234, 3.4e38
	double	64	14.0, 1.234, 4.9e-324
Olioviitteet	LuokanNimi		

Javassa erikoisasemassa olevat olioluokat: taulukot ja String, joille on omaa syntaksia itse kielessä. Esim.:

	Tyyppi	alustusliteraali
Taulukko	int[]	{1, 2, -7}
Merkkijono	String	"Heippa"

Siinäkö oli kaikki tietorakenteet?

- Primitiivityypit, merkkijonot ja taulukot on toteutettu useimmissa ohjelmointikielissä valmiina (eli kieli tarjoaa keinot eikä tarvitse itse ohjelmoida tietorakenteen toteutusta). Tulemme näkemään, millä tavoin tämä on tehty Javassa.
- Usein käytetään monipuolisempia tietorakenteita (esim. pinot, jonot, listat, puut, hakemistot, verkot, . . .) mutta ne pitää yleensä toteuttaa käyttäen yksinkertaisempia, ohjelmointikielen tarjoamia rakenteita. Miten? Luomalla sisäinen toteutus ohjelmointikielen keinoin ja tarjoamalla aliohjelmien/metodien kautta käytettävä rajapinta.

Oppikirjaviitteitä

Mistä kohtaa oppikirjaa (Mika Vesterholm ja Jorma Kyppö: Java-ohjelmointi, 6., uudistettu painos) löytyy näitä samoja asioita:

- Lopulta sivut 174-200, luku ”7 Merkit ja merkkijonot”. Tässä vaiheessa noin sivut 174-181.
- Lopulta sivut 201-215, luku ”8 Taulukot”. Tässä vaiheessa n. sivut 201-205.
- Omien olioluokkien tekemisestä kertovat kohdat eivät (ainakaan vielä) ole kurssin keskiössä, vaan Javan syntaktiset elementit, tietotyypit ja perusohjelman rakenne.
- Terminologia ja lähestymistapa on kirjassa hieman eri kuin luennoilla. Pohjimmiltaan kyse on samoista asioista!